



An adaptive large neighborhood search heuristic for the planar storage location assignment problem: application to stowage planning for Roll-on Roll-off ships

Jone R. Hansen¹ · Kjetil Fagerholt¹ · Magnus Stålhane¹ · Jørgen G. Rakke²

Received: 26 September 2019 / Revised: 15 July 2020 / Accepted: 17 July 2020
© The Author(s) 2020

Abstract

This paper considers a generalized version of the planar storage location problem arising in the stowage planning for Roll-on/Roll-off ships. A ship is set to sail along a predefined voyage where given cargoes are to be transported between different port pairs along the voyage. We aim at determining the optimal stowage plan for the vehicles stored on a deck of the ship so that the time spent moving vehicles to enable loading or unloading of other vehicles (shifting), is minimized. We propose a novel mixed integer programming model for the problem, considering both the stowage and shifting aspect of the problem. An adaptive large neighborhood search (ALNS) heuristic with several new destroy and repair operators is developed. We further show how the shifting cost can be effectively evaluated using Dijkstra's algorithm by transforming the stowage plan into a network graph. The computational results show that the ALNS heuristic provides high quality solutions to realistic test instances.

Keywords Maritime transportation · Packing · Stowage · Roll-on/Roll-off · RoRo

1 Introduction

This paper considers a generalized version of the planar storage location assignment problem (PSLAP), which is a sub-class of the general storage location assignment problem (SLAP). The SLAP is to assign incoming products to storage locations in storage departments/zones in order to reduce material handling cost and improve space utilization (Gu et al. 2007). This problem arises in many industrial applications, e.g. warehouse management (Xiao and Zheng 2010), and container terminal operations

✉ Jone R. Hansen
jone.hansen@ntnu.no

¹ Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Trondheim, Norway

² Wallenius Wilhelmsen Logistics, Lysaker, Norway

(Chen et al. 2003). In warehouse operations, a warehouse typically has one entry point where the incoming products arrive before they are transported to a storage zone and put in a shelf. When there is a demand for a product, it is retrieved from its storage location and delivered to an exit point. For a recent survey on SLAP we refer to Reyes et al. (2019).

The PSLAP, first introduced by Park and Seo (2009), considers the storage of incoming objects on a planar, two-dimensional surface. When retrieving a given object from its location, it may be necessary to temporarily move other objects in order to retrieve it, as other objects may block every possible retrieval route. The objective is to minimize the number of such undesirable relocations. Park and Seo (2009) consider a version of the PSLAP arising in the assembly block stockyard operations at a shipyard, where the stored and retrieved objects are multiple assembly blocks. These blocks are moved into the yard for welding, outfitting, and painting at given times. Finally, they are moved to the dry dock for final assembly. It is assumed that assembly blocks are stored into the yard and retrieved from the yard in a straight line. While temporary relocations may be unavoidable due to limited storage areas on the yard, it is clear that the number of relocations should be kept at a minimum to reduce costs. A genetic algorithm for the PSLAP with applications to assembly block stockyard operations is proposed by Park and Seo (2010), while Tao et al. (2013) present a tabu search heuristic for the same problem.

In this paper we consider a generalized version of the PSLAP arising in the stowage planning for Roll-on/Roll-off (RoRo) ships. RoRo-ships have multiple decks on which various rolling material, such as cars, trucks, and heavy machinery, are transported. It is usually decided at a higher planning level which cargoes to be placed on which deck, considering stability and deck characteristics. We therefore study the two-dimensional RoRo ship stowage problem (2DRSSP) for a single deck.

The 2DRSSP was first introduced by Hansen et al. (2016) and is the problem of determining a stowage plan, given a predefined voyage, visiting several ports. At each port, a given set of cargoes are to be loaded onto or unloaded from the ship through a specific entry/exit point. Each cargo consists of a given number of identical vehicles and has a predetermined loading and unloading port. The 2DRSSP is to assign vehicles (incoming products) to locations on the deck (storage zones) in order to minimize the time used on shifting vehicles (undesirable relocations), which means temporarily moving some vehicles to enable loading or unloading of other vehicles. This can be a time-consuming procedure, as the vehicles usually are attached to the deck using chains during the transport.

There exist also a few other studies considering optimization of stowage plans for RoRo-ships. Øvstebø et al. (2011a) propose a mixed integer programming (MIP) model and a heuristic method for solving this problem. In contrast to the 2DRSSP studied in this paper, they do not have to transport all cargoes, so the objective is to maximize the sum of revenue from optional cargoes, minus the penalty costs incurred from shifting. For modeling purposes, Øvstebø et al. (2011a) divide the decks into several logical lanes into which the vehicles are lined. Dividing the decks into lanes might be a reasonable assumption when the cargoes are very similar in size (e.g. only regular cars). However, it can be very limiting in many other practical situations, such as for more general RoRo-ships which carry everything from cars, via trucks, military

and agricultural equipment, to breakbulk cargoes, as these cargoes can be very different in size and shape. Dividing the deck into lanes with the same width would therefore give poor capacity utilization. Therefore, the model that we propose here does not rely upon this assumption. Øvstebø et al. (2011b) extend the work from Øvstebø et al. (2011a) by also considering the routing and scheduling of the RoRo-ships, though relying on the same limiting modeling assumption on dividing the deck into lanes.

Puisa (2018) introduces a mathematical model for RoRo-ship stowage planning, which considers multiple decks, fire safety, and ship stability. However, the model has several limitations compared with the model presented in this paper. First, each deck is divided into a separate grid for each type of cargo, which is computationally desirable, but may lead to poor utilization of the deck when cargoes have different sizes and shapes. Second, shifting of cargoes in port during a voyage is not penalized in the model, but either forbidden (which likely leads to less cargo stowed on-board the vessel), or allowed at zero cost (which may lead to prohibitively long port stays). Further, no solution method is presented except solving the model using a commercial solver, which is unlikely to be computationally tractable for the larger RoRo-ships considered in this paper.

While the 2DRSSP is categorized as a SLAP (and PSLAP), it is also closely related to the two-dimensional Single Large Object Placement Problem (2SLOPP). In the 2SLOPP, a weakly heterogeneous set of small items (vehicles) are placed onto a large item (deck), to maximize the total value of the items placed. This problem arises in many industrial applications concerned with the cutting of materials such as steel, glass, wood, and textile, see for example Haims and Freeman (1970) and Wang (1983)). For 2DRSSP addressed in this paper, we do not aim to maximize the total value of the items packed, but rather constrain the solution space to ensure that all items are packed. Instead, the objective is dependent on the items' placements relative to one another, i.e. shifting.

Packing problems appear also in other maritime applications, though differing from the 2DRSSP. Bayliss et al. (2019) propose a heuristic approach for solving the vehicle ferry revenue management problem with dynamic pricing. A packing algorithm, which uses simulated annealing to search for the set of parameters maximizing the packing efficiency, is used to optimize the stowage. However, since the ferry sails between only two ports, they do not have to deal with shifting. Seixas et al. (2016) present a heuristic for a pick-up and delivery allocation problem for platform supply vessels, where they aim to determine the optimal cargo allocation on deck. In contrast to the 2DRSSP, they do not have to carry all cargoes, so their problem is modeled as two-dimensional knapsack problem. Furthermore, they do not consider shifting as the cargoes can be accessed with a crane from above.

Yet another application of packing in maritime transportation is the stowage planning of container ships, see for example (Wei-Ying et al. 2005; Tierney et al. 2014; Ding and Chou 2015), as well as the survey by Iris and Pacino (2015). In this problem, containers are packed in three dimensions, and one sometimes needs to temporarily move (shift) containers to get access to containers that are placed below. In this case, determining the number of shifts to (un)load a container, given a stowage plan, is done simply by counting the containers on top of it. Container stowage is not only relevant for the shipping company, but also for the terminal operators, see for example the

studies by Monaco et al. (2014) and Iris et al. (2018) which consider the integrated container stowage problem.

Our main contributions are twofold. First, we introduce a novel mixed integer programming (MIP) model for the 2DRSSP, which considers both stowage and shifting. Second, we present an Adaptive Large Neighborhood Search (ALNS) heuristic for solving the 2DRSSP and demonstrate through a computational study its ability to construct practically implementable stowage plans. The heuristic makes use of several problem-specific destroy and repair operators, which are new to the literature. While the problem is formulated with respect to its industrial application, the proposed solution method may be applied to PSLAPs arising in other industries, such as the aforementioned assembly block stockyard operations, vehicle yard management (Cordeau et al. 2011; Mattfeld and Orth 2006), and general cargo storage (e.g. unstackable pallets).

The remainder of the paper is organized as follows. We define the stowage problem and present mathematical formulations in Sect. 2. In Sect. 3 we present an adaptive large neighborhood search heuristic for solving the practical problem. Computational results are presented in Sect. 4, before concluding remarks are drawn in Sect. 5.

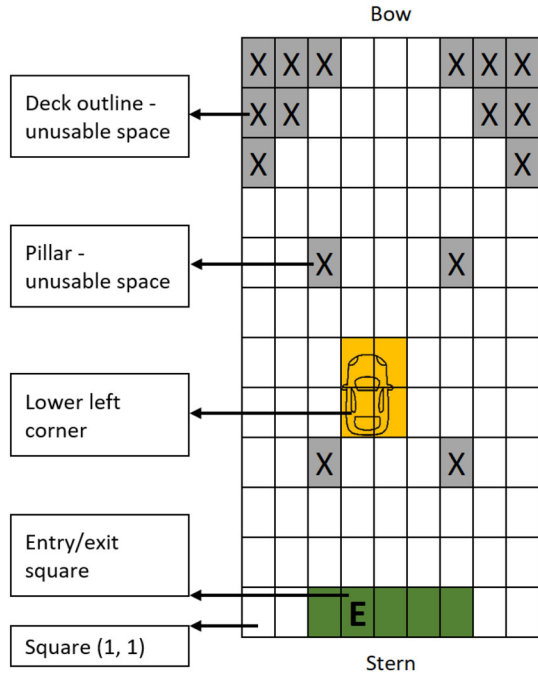
2 Problem definition and mathematical formulations

In this section, we give a formal description of the two-dimensional RoRo ship stowage problem (2DRSSP) with shifting cost, which is an extension of the stowage model presented by Hansen et al. (2016). In their work, heuristic rules are used to guide the placement of the vehicles. To determine the true objective value of a solution, an additional optimization problem must be solved, namely the stowage plan evaluation problem (SPEP) introduced by Hansen et al. (2017). In this work, we extend the original stowage model defined by Hansen et al. (2016) to account for shifting explicitly, building on the ideas of Hansen et al. (2017).

A voyage is defined by a sequence of port calls, where \mathcal{P} is the set of all ports, and \mathcal{C} is the set of cargoes (or shipments) to be transported along the voyage. Each cargo $c \in \mathcal{C}$ is to be loaded at port P_c^L and unloaded at port P_c^U . A cargo c consists of R_c identical vehicles (or units) with the same length, width, and weight, denoted V_c^L , V_c^W , and V_c^M , respectively. There is a minimum required clearance between the vehicles when stowed on the deck defined by the parameter B . It is assumed that all cargoes are mandatory to transport due to contractual terms. The deck has a given length D^L and width D^W and unusable stowage spaces such as pillars, safety areas, and ramps, are given. The objective is to place all mandatory cargoes on the deck in such a way that the shifting cost of the stowage plan is minimized. The shifting cost reflects the costs and/or time used to temporarily move vehicles to enable loading and unloading of other vehicles.

The model presented below is based on a grid representation of the deck, where \mathcal{I} is the set of rows and \mathcal{J} the set of columns, illustrated by the toy example in Fig. 1. A square (i, j) represents a physical area on the deck where the vehicles may be placed, where square $(1, 1)$ is defined as the square located at the stern of the ship's port side. All vehicles are modeled as rectangular objects. Thus, the number

Fig. 1 Illustration of the modeling approach, highlighting the main points



of squares needed to place a vehicle from cargo c in length and width is given by $S_c^L = \lceil (V_c^L + B) | \mathcal{I} / D^L \rceil$ and $S_c^W = \lceil (V_c^W + B) | \mathcal{J} / D^W \rceil$ in the longitudinal and latitudinal directions, respectively. The positioning of each vehicle is given by the location of its lower left corner. Thus, if a vehicle from cargo c is placed (with its lower left corner) in square (i, j) , it spans over all squares $(i', j') \in i' = i \dots i + S_c^L - 1, j' = j \dots j + S_c^W - 1$. We see from Fig. 1 that the vehicle placed with its lower left corner in square $(5, 4)$ also covers squares $(5, 5)$, $(6, 4)$, and $(6, 5)$. Using this approach, we overestimate the area usage of the vehicles. However, with sufficiently high grid resolution, this overestimation becomes negligible. We refer to Hansen et al. (2016) for a more detailed and illustrative description of the modeling approach.

Let $\mathcal{N} \subset \mathcal{I} \times \mathcal{J}$ be the set of all squares (i, j) where vehicles may be placed, i.e. not unusable spaces such as pillars, safety areas, and ramps. Further, let $\mathcal{N}_c \subseteq \mathcal{N}$ be the set of squares where the lower left corner of a vehicle from cargo c may be placed, where both unusable areas and weight limitations are accounted for. Let Q_{ijc} be the set of squares where placing a vehicle from cargo c would cover the square (i, j) . For example, if a cargo c consists of 2×2 sized vehicles, then $Q_{ijc} = \{(i - 1, j - 1), (i - 1, j), (i, j - 1), (i, j)\}$, as a vehicle from cargo c placed in any of these squares would cover square (i, j) .

We define squares placed next to square (i, j) as neighboring squares, given by the set $\mathcal{B}_{ij} = \{(i + 1, j), (i - 1, j), (i, j + 1), (i, j - 1)\}$. Let the set of arcs $\mathcal{A}_c \subset \mathcal{N}_c \times \mathcal{N}_c$ define the feasible movements for the vehicles in cargo c . Let \mathcal{C}_p^R be the set of cargoes that is to be loaded or unloaded at port p . Further, let \mathcal{C}_p^N be the set of cargoes that is placed on the ship at port p , given that the cargoes neither is to be loaded nor unloaded

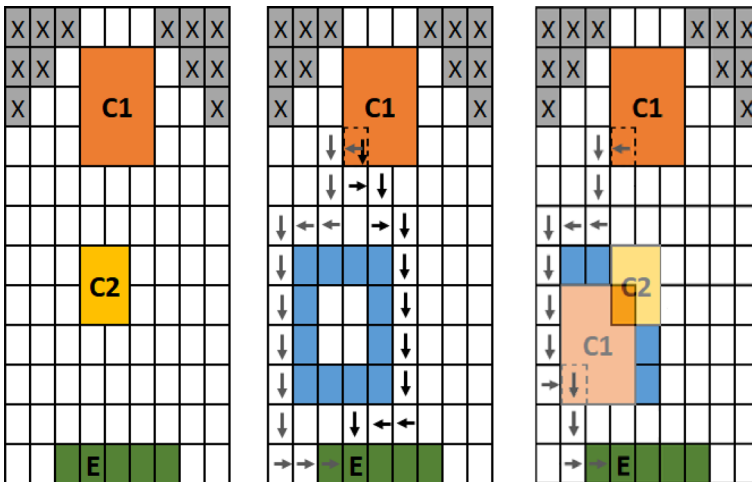


Fig. 2 To the left, the stowage plan at a given port is shown, where C1 and C2 are vehicles. A vehicle in cargo 1 (C1) is to be unloaded at this port. If the lower left corner of the vehicle from C1 (square marked with a dashed line) uses any of the blue squares when unloading, it will initiate a shift of the vehicle from C2. In the middle, two unloading routes that do not require any shift, are given. The location of the lower left corner is tracked. To the right, an example of an unloading route that requires shifting is given. As the lower left corner of vehicle C1 uses a blue square when unloading and hence, the vehicles overlap, vehicle C2 must be shifted (Color figure online)

at port p . If a vehicle from cargo $c \in \mathcal{C}_p^N$ is shifted at port p , a shifting cost C_c^M per unit of cargo c is imposed. The shifting cost is based on the vehicle’s area, as it is usually more time-consuming to move larger vehicles than smaller ones. It is assumed that a shifted vehicle is moved off the deck during the port call and returned to the same location after the loading/unloading.

Let $\mathcal{P}^{LU} = \mathcal{P} \setminus \{1, |\mathcal{P}|\}$ be the set of ports where shifting may occur, i.e. every port except the first and the last port along the voyage, since no vehicles are shifted in these ports. Further, let $\mathcal{P}_c^{LU} = \{P_c^L, P_c^U\}$ be the set of the loading and unloading port of cargo c . Parameters E^I and E^J give the location of the entry/exit square on the deck, respectively.

Let \mathcal{D}_{ijcd} be the set of all squares that would initiate a shift of a vehicle from cargo d , placed in square (i, j) , if the lower left corner of a vehicle from cargo c uses any of these squares when it is loaded or unloaded, illustrated in Fig. 2. In this example, a vehicle from cargo C2 is placed with its lower left corner in row five, column four, i.e. square $(5, 4)$. Thus, the set \mathcal{D}_{54C1C2} contains all the squares marked in blue in Fig. 2, as using these squares when routing C1 from its initial position to the exit would initiate a shift of the vehicle from C2. Note that the squares inside the outer edge of the blue rectangle are not added to the set, as the vehicle from C1 would already have used at least one of the blue squares in order to reach the inner squares.

As in Hansen et al. (2016), the following practical assumptions are made regarding the problem. First, the ports are assumed to be separated into a loading region and an unloading region, which is how most voyages are in RoRo-shipping. Next, it is assumed that once a vehicle is placed, it stays in the same location during its time on

the vessel. A vehicle is always placed longitudinal to the deck, i.e. with its front facing the bow, which is most common. Given these assumptions, all vehicles are placed on the deck on the sailing leg between the last loading port and the first unloading port. Thus, we only need to construct a stowage plan for this sailing leg, as a stowage plan for all other sailing legs can be derived from it.

Let the binary decision variables x_{ijc} denote whether the lower left corner of a vehicle from cargo c is placed in square (i, j) , while the binary decision variables δ_{ijcp} is equal to 1 if this vehicle is shifted at port p , and 0 otherwise. The arc flow variable f_{ijklcp} represents the flow of vehicles from cargo c from square (i, j) to a neighboring square (k, l) at port p . In Fig. 2, the arrows represent these flow variables being equal to one. The decision variable d_{ijcp} represents the supply/demand for cargo c at square (i, j) in port p . If $d_{ijcp} > 0$, square (i, j) is a supply square for cargo c at port p ; if $d_{ijcp} < 0$ square (i, j) is a demand square for cargo c at port p with a demand of $-d_{ijcp}$; and if $d_{ijcp} = 0$, square (i, j) is a transit square for cargo c at port p . For example, if cargo c consists of five vehicles, then there is a supply of five vehicles from cargo c at the entry square in the loading port of cargo c , i.e. $d_{E^I E^J c P_c^L} = 5$. Further, for each of the five squares the vehicles are placed, i.e. $x_{ijc} = 1$, there is a demand for a vehicle from cargo c at its loading port, $d_{ijc P_c^L} = -1$. All other squares have zero supply/demand for cargo c .

The combined stowage and shifting model can be formulated as follows:

$$\min z = \sum_{c \in \mathcal{C}} \sum_{(i,j) \in \mathcal{N}_c} \sum_{p \in \mathcal{P}^{LU}} C_c^M \delta_{ijcp} \tag{1}$$

subject to:

$$\sum_{(i,j) \in \mathcal{N}_c} x_{ijc} = R_c, \quad c \in \mathcal{C} \tag{2}$$

$$\sum_{c \in \mathcal{C}} \sum_{(i',j') \in \mathcal{Q}_{ijc}} x_{i'j'c} \leq 1, \quad (i, j) \in \mathcal{N} \tag{3}$$

$$\delta_{ijcp} \leq x_{ijc}, \quad p \in \mathcal{P}^{LU}, c \in \mathcal{C}_p^N, (i, j) \in \mathcal{N}_c \tag{4}$$

$$d_{E^I E^J c P_c^L} = R_c \quad c \in \mathcal{C} \tag{5}$$

$$d_{ijc P_c^L} = -x_{ijc} \quad c \in \mathcal{C}, (i, j) \in \mathcal{N}_c \setminus \{(E^I, E^J)\} \tag{6}$$

$$d_{E^I E^J c P_c^U} = -R_c \quad c \in \mathcal{C} \tag{7}$$

$$d_{ijc P_c^U} = x_{ijc} \quad c \in \mathcal{C}, (i, j) \in \mathcal{N}_c \setminus \{(E^I, E^J)\} \tag{8}$$

$$\sum_{(k,l) \in \mathcal{B}_{ij}} f_{ijklcp} - \sum_{(k,l) \in \mathcal{B}_{ij}} f_{klijcp} = d_{ijcp} \quad p \in \mathcal{P}^{LU}, c \in \mathcal{C}_p^R, (i, j) \in \mathcal{N}_c \tag{9}$$

$$\sum_{c \in \mathcal{C}_p^R} \sum_{(k',l') \in \mathcal{D}_{klcd}} \sum_{(i,j) \in \mathcal{B}_{k'l'}} (f_{ijk'l'cp} + f_{k'l'ijcp}) \leq M_{dp}(1 - x_{klc} + \delta_{klcdp}) \quad p \in \mathcal{P}^{LU}, d \in \mathcal{C}_p^N, (k, l) \in \mathcal{N}_d \tag{10}$$

$$x_{ijc} \in \{0, 1\}, \quad c \in \mathcal{C}, (i, j) \in \mathcal{N}_c \tag{11}$$

$$\delta_{ijcp} \in \{0, 1\}, \quad c \in \mathcal{C}, (i, j) \in \mathcal{N}_c, p \in \mathcal{P}_c^{LU} \tag{12}$$

$$d_{ijcp} \in \{0, 1\}, \quad c \in \mathcal{C}, (i, j) \in \mathcal{N}_c \setminus \{(E^I, E^J)\}, p \in \mathcal{P}_c^{LU} \tag{13}$$

$$d_{E^i E^j c p} \in \mathbb{R}, \quad c \in \mathcal{C}, p \in \mathcal{P}_c^{LU} \tag{14}$$

$$f_{ijklcp} \geq 0 \quad p \in \mathcal{P}^{LU}, c \in \mathcal{C}_p^R, ((i, j), (k, l)) \in \mathcal{A}_c \tag{15}$$

The objective function (1) minimizes the shifting cost. Constraints (2) ensure that all vehicles in all cargoes are placed on the deck. Constraints (3) guarantee that each square is used by at most one vehicle. Constraints (4) guarantee that a vehicle from cargo c may only be shifted from square (i, j) at port p if it is placed in square (i, j) . Constraints (5) set the supply of vehicles from cargo c at the entry square equal to the number of vehicles in cargo c at the cargo’s loading port. Constraints (6) ensure that if a vehicle from cargo c is placed in square (i, j) , there is a demand at this square at the cargo’s loading port. If no vehicle from cargo c is placed in square (i, j) , the demand is zero, which makes it a transit square. Constraints (7)–(8) state the demand and supply for the unloading port of cargo c , similar to constraints (5)–(6). The flow balance constraints (9) state that the outflow minus inflow must equal the supply/demand of the square (i, j) for each cargo c that is loaded or unloaded at port p . Constraints (10) state that for each port p , if any vehicle use a square that initiates a shift of a vehicle from cargo d placed in (k, l) on its loading or unloading route, the blocking vehicle from cargo d placed in (k, l) must be shifted. Finally, variable declarations are given by constraints (11)–(15).

3 Adaptive large neighborhood search heuristic

For small instances, the combined stowage and shifting model for the 2DRSSP can be solved to optimality by a commercial MIP solver. To solve realistically sized instances, we propose a new adaptive large neighborhood search (ALNS) heuristic. The ALNS heuristic was introduced by Ropke and Pisinger (2006). Given an initial solution, the heuristic sequentially destroys and repairs the solution in order to improve the objective and is commonly used to solve routing problems, see for example Ribeiro and Laporte (2012) and Hemmelmayr et al. (2012). In the maritime domain, ALNS heuristics have been used to solve two-dimensional allocation problems, see for example Mauri et al. (2016) and Iris et al. (2017). However, to the authors’ knowledge, ALNS has not previously been applied to stowage problems, such as the 2DRSSP.

3.1 Heuristic overview

The pseudocode for the heuristic is provided in Algorithm 1. An initial solution \mathcal{X}^0 is generated by a greedy randomized adaptive search procedure (GRASP), described in Sect. 3.2. We define solution \mathcal{X} as the current solution and solution \mathcal{X}^* as the best solution found. At the beginning of each iteration, a temporary stowage plan \mathcal{X}' is created, which is a copy of the current solution. Solution \mathcal{X}' is then partially destroyed by a destroy operator d and a repair operator r attempts to repair the solution. If successful, the quality of the stowage plan \mathcal{X}' is estimated using the functions $G(\mathcal{X}')$ and $P(\mathcal{X}')$. If the stowage plan seems promising, we calculate the shifting cost $c(\mathcal{X}')$ of the stowage plan. These estimation functions are included to reduce the time spent on

evaluating unpromising solutions, as the shifting cost evaluation is a time-consuming procedure. We discuss the estimation and evaluation of the shifting cost in more detail in Sect. 3.5. The shifting cost $c(\mathcal{X}')$ is then compared with the shifting cost of the current solution and scores are awarded accordingly. Every M iterations, the operators' weights are updated based on the awarded scores (see Sect. 3.6). The search is stopped when a solution with zero shifting cost is found or the computing time limit is reached, and the best solution \mathcal{X}^* is returned.

Algorithm 1 Pseudocode for the 2DRSSP ALNS

```

1: Construct an initial solution via GRASP  $\mathcal{X}^0$ ;  $\mathcal{X} \leftarrow \mathcal{X}^0$ ;  $\mathcal{X}^* \leftarrow \mathcal{X}^0$ 
2: Initialize weights:  $w^{dest}, w^{rep}$ 
3: Set segment size =  $M$ , iteration counter  $i = 0$ 
4: while until stopping criteria met do
5:    $\mathcal{X}' \leftarrow \mathcal{X}$ 
6:   Select destroy and repair operators  $d \in \mathcal{D}, r \in \mathcal{R}$  using  $w^{dest}, w^{rep}$ 
7:   Apply destroy operator  $d$  to  $\mathcal{X}'$ 
8:   Apply repair operator  $r$  to  $\mathcal{X}'$ 
9:   if feasible( $\mathcal{X}'$ ) and  $(P(\mathcal{X}')(1+p) \geq P(\mathcal{X})$ ) or  $G(\mathcal{X}')(1+g) \geq G(\mathcal{X})$  then
10:     Calculate shifting cost of  $\mathcal{X}', c(\mathcal{X}')$ 
11:     if  $c(\mathcal{X}') < c(\mathcal{X}^*)$  then
12:        $\mathcal{X}^* \leftarrow \mathcal{X}'$ 
13:        $\mathcal{X} \leftarrow \mathcal{X}'$ , award score  $\sigma_1$  to the operators
14:     end if
15:   end if
16:   if  $i \% M = 0$  then
17:     Update weights  $w^{dest}, w^{rep}$ 
18:   end if
19: end while
20: return  $\mathcal{X}^*$ 

```

3.2 Construction of an initial solution

The first step of the ALNS heuristic is to construct a feasible initial solution. In order for a solution to be feasible, all vehicles in every cargo must be placed on the deck. We propose using a greedy randomized adaptive search procedure (GRASP) for constructing an initial solution. First, we give a general presentation of the GRASP. Second, we explain how the GRASP is implemented for this stowage problem and explain any modification done to the general framework.

The GRASP was initially proposed by Feo and Resende (1989). It is a metaheuristic, in which each iteration consists of two phases: Construction and local search (Resende and Ribeiro 2014). The construction phase builds a solution iteratively. In each iteration, one element is added to the solution. The element is randomly selected from a restricted candidate list (RCL) of elements, where the RCL contains the l best candidates to be added to the solution. The list is updated after every iteration. By selecting a random element, and not necessarily the best one in each iteration, the procedure allows for different solutions to be constructed. The length of the list defines the degree of randomization. With increasing RCL length, diversification is added to the search due to the increasing possible element choices. When a solution is constructed,

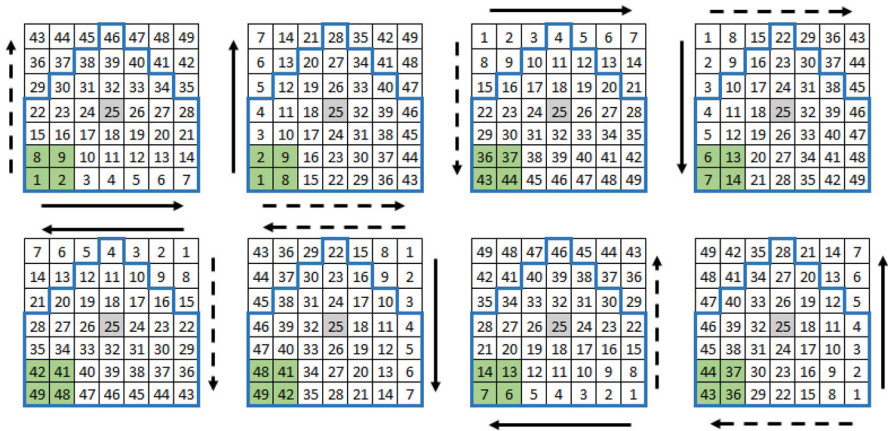


Fig. 3 The eight square sequences. Squares are evaluated according to the numbering in ascending order. The solid arrow gives the primary direction, while the dashed line shows the secondary direction. Any squares outside the deck outline or unusable squares are not evaluated but are numbered here for illustrative purposes

a local search algorithm is applied to improve the solution. If it is a new best solution, the solution is kept. The procedure of constructing a solution and improving it is then repeated until a stopping criterion is met. We refer to Feo and Resende (1995) for a more detailed explanation of the GRASP.

In this work, we use the ALNS for improving the solutions. Hence, we only use the GRASP for constructing a feasible solution, and the use of a local search algorithm is omitted. We construct the solutions from an empty deck, by repeatedly placing vehicles on the deck until all vehicles are placed (success), or none of the remaining vehicles can be placed (failure). In each step, two decisions must be made: (1) Which square should we place the next vehicle on, and (2) which vehicle should we place on the selected square. The latter decision is handled by the RCL. The list consists of the l best insertion elements, i.e. the best cargoes to assign a given location on the deck. The list length is implemented as a self-tuning parameter, which means that the probability of selecting a given list length depends on the previously obtained solutions for the same list length. This reactive feature has yielded promising results in other work (see e.g. Prais and Ribeiro 2000), while also reducing the complexity of the parameter setting as it self-tunes.

The decision of which square to evaluate next is handled by the square sequence parameter. A given square sequence d is an ordered list of squares, which determines the next square (i, j) to evaluate. Eight square sequences are predefined, that is two directions for each of the four corners of the deck layout, illustrated in Fig. 3. The reasoning for using fixed square sequences is that one should expect the resulting stowage plan to have less unused squares between the vehicles, compared to randomly drawing the next square, which is likely to increase the success rate of the construction phase. The RCL length and the square sequence parameters are referred to as search guiding parameters (SGP).

A pseudocode for the GRASP is presented in Algorithm 2. In line 2, the initial weights of the SGPs are set. Let vectors \mathbf{w}^D and \mathbf{w}^L give the probabilities of selecting each possible value of the corresponding SGPs, i.e. square sequence and RCL length, respectively. For example, for the square sequence SGP, the vector \mathbf{w}^D consists of eight elements, i.e. one element for each square sequence. In line 3, the RCL length and square sequence for the first iteration is set, using a roulette wheel selection method (RWS) based on the vectors \mathbf{w}^D and \mathbf{w}^L . For the RWS, the probability p_d^D of selecting square sequence d is then calculated as $p_d^D = w_d^D / \sum_{d \in \mathcal{D}} w_d^D$, where \mathcal{D} is the set of square sequences. The same procedure is used for \mathbf{w}^L .

Algorithm 2 Pseudocode for the 2DRSSP GRASP

```

1: procedure GRASP
2:   Initialize weights:  $\mathbf{w}^D, \mathbf{w}^L$ 
3:    $d = \text{RWS}(\mathbf{w}^D), l = \text{RWS}(\mathbf{w}^L)$ 
4:   while stopping criteria not met do
5:      $\mathcal{S} = \mathcal{N}$ , i.e. set of all usable squares
6:      $\mathcal{X} = \emptyset$ 
7:     Randomly select a SGP, and update the parameter using RWS
8:     Sort( $\mathcal{S}$ ) according to square sequence  $d$ 
9:     while  $\mathcal{S} \neq \emptyset$  do
10:       $(i, j) = \text{first element of } \mathcal{S}$ . Remove this element from  $\mathcal{S}$ 
11:       $\text{RCL} = \text{makeRCL}((i, j), l)$ 
12:       $x_{ijc}^* = \text{random}(\text{RCL})$ 
13:       $\mathcal{X} = \mathcal{X} \cup \{x_{ijc}^*\}$ 
14:     end while
15:     if AllVehiclesPlacedOnDeck( $\mathcal{X}$ ) = true then
16:       return  $\mathcal{X}$ 
17:     end if
18:     UpdateWeights( $w_d, w_l, w_r, w_o, \delta$ )
19:   end while
20: end procedure

```

In lines 9–14, the procedure attempts to build a feasible stowage plan. A list of squares, sorted according to the selected square sequence d is given as input. For the square selected in line 10, a RCL is constructed in line 11. All cargoes that have a vehicle which may be placed in square (i, j) are added to the RCL. For a cargo c to be in the RCL, two conditions must be satisfied: (1) All vehicles in the cargo are not yet placed on the deck, and (2) it is possible to insert a vehicle from cargo c in the evaluated square (i, j) . This is possible if $(i, j) \in \mathcal{N}_c$ and no other vehicles use any square $(i', j') \in \mathcal{P}_{ijc}$, where \mathcal{P}_{ijc} includes all squares used by a vehicle from cargo c , if placed in square (i, j) . These squares are often available as the GRASP inserts vehicles using predefined square sequences.

Before reducing the RCL to the desired length l , defined by the RCL length SGP, we sort the list to ensure the best possible insertions remain in the RCL. We identify the best candidates based on the vehicles’ area usage. By inserting the largest vehicles first, more flexibility is given in the later stages of the construction phase. Next, the RCL is reduced to length l . In line 12, a cargo is sampled from this list and a vehicle

from this cargo is added to the stowage plan in line 13. This element refers to the variable x_{ijc} being equal to 1, for the drawn cargo from the RCL. The RCL may be empty, meaning that no vehicles from the carried cargoes can be placed in square (i, j) . Which corner of the vehicle that is placed in the square (i, j) is determined in the following manner: If a square sequence starting in the lower left corner is used, the lower left corner of the vehicle is placed in the evaluated square. Following this logical structure, the corresponding corner of the start square of the square sequence is used when placing the vehicles. Unless otherwise is stated, the default square sequence is set to start in the lower left corner and is used without further explanation in the following sections.

When all squares are evaluated, a feasibility test of \mathcal{X} is performed. If all vehicles are placed on the deck, the stowage plan is feasible, and the algorithm terminates. If the feasibility test fails, a new iteration of the GRASP starts. After every iteration, the weights for each SPG vector are updated in line 18. The weights are updated according to how well the selected SPGs performed in the current iteration, compared to the previous iteration, where the sum of the unplaced vehicles' area is compared. Thus, if the sum of the unplaced vehicles' area in the current stowage plan is less than the previous stowage plan, the updated parameter is rewarded a positive score δ , implying that this parameter choice increases the probability of creating a feasible stowage plan. For the opposite outcome, a score of $-\delta$ is imposed to the parameter's weight. If the GRASP fails to construct a feasible solution within the time limit, the ALNS procedure terminates. Another alternative is to limit the time used on the GRASP and continue to the destroy and repair phase without an initial solution. The repair operators could be used to search for a feasible solution from an empty solution until the time limit is reached. However, preliminary testing has shown that the GRASP heuristic is the better method for constructing feasible solutions from an empty deck. As such, we let the GRASP run until a feasible solution is found, or the computing time limit is reached.

Two minor modifications are made to the GRASP when implementing the heuristic. In the first eight iterations, each square sequence is used once and the RCL length is set to one. Preliminary testing has shown that this greedy search often provides promising solutions, given a successful feasibility test. Additionally, in the first eight iterations, the search is not terminated if a feasible solution is found. If more than one feasible solution is found, we evaluate the shifting cost of each solution, and the solution with the lowest shifting cost is used as the initial solution. This is done to give the ALNS a better starting point.

3.3 Destroy operators

This section presents the six different destroy operators that are used, where the majority are motivated by some problem-specific characteristics. The destroy parameter ξ , $0 < \xi \leq 1$, determines the percentage of vehicles removed from the solution. All removed vehicles are inserted in the set \mathcal{L} .

3.3.1 Neighbor removal

This operator removes vehicles that do not have a neighboring vehicle from the same cargo. A vehicle placed in square (i, j) has a neighbor if a vehicle from the same cargo is placed in any of the squares $(i + S_c^L, j)$, $(i - S_c^L, j)$, $(i, j + S_c^W)$, or $(i, j - S_c^W)$. Let λ be the share of vehicles without a neighbor. If $\lambda < \xi$, all vehicles without a neighbor is removed. If $\lambda \geq \xi$, a random selection of the vehicles without neighbor is done. The purpose of this operator is to increase the number of neighboring vehicles, which is assumed to reduce the shifting cost of the stowage plan.

3.3.2 Area removal

The operator removes all vehicles placed within a certain area of the deck. The size of the area is given by the destroy parameter ξ , whereas the area's shape (length and width) and location are randomly selected. By removing all vehicles from a certain area, the reinsertion procedures try to improve the vehicles' placement on a subsegment of the deck, which could reduce the overall shifting cost of the stowage plan.

3.3.3 Random removal

The Random removal operator simply removes ξ share of the vehicles from the deck, randomly selected. The inclusion of this operator is purely based on the fact that it induces diversification to the search.

3.3.4 Port removal

This destroy operator removes vehicles according to their loading and unloading port. The procedure is as follows: (1) select a port at random. (2) Until ξ share of all vehicles on deck are removed, randomly remove a vehicle loaded or unloaded at the given port. If all vehicles to be loaded or unloaded at the given port are removed, go to (1), else end procedure.

3.3.5 Shifting cost removal

The shifting cost removal operator removes vehicles based on the shifting evaluation results of the current stowage plan. The solution from this evaluation gives information regarding which vehicles that are shifted most often during the voyage. Rearranging the placement of these vehicles seems reasonable as they are located in areas where many of the vehicles pass through when loading/unloading at ports. We combine this idea with the area removal, to enable these vehicles to be moved away from their current location. Thus, the top $\xi/2$ share of the vehicles with regards to shifting cost is removed, in addition to vehicles within an area of $\xi/2$ of the deck size, i.e. area removal.

3.3.6 Route removal

The route removal operates similarly to the shifting cost removal. However, instead of removing the vehicles with the highest shifting cost, it removes the vehicles that are most “costly” to load/unload. Given the solution from the SPEP of the current stowage plan, we can calculate the cost of loading and unloading each vehicle by summing the shifting costs of all blocking vehicles. For example, if a vehicle loaded at the last loading port is placed at the innermost location on the deck, we expect this to cause a lot of shifting. This operator aims to correct such decisions, which should be a good basis for repairing solutions. As for the shifting cost removal, $\xi/2$ of the destruction is allocated to the area removal, while the remaining share is used for the route removal.

3.4 Repair operators

The vehicles in the removal list are attempted reinserted by one of four repair heuristics. If all vehicles are successfully reinserted, the repaired solution is evaluated. If the selected repair operator fails to repair the solution, the shifting cost evaluation is skipped, and a new iteration of the ALNS starts. The repair operators are briefly described in the following.

3.4.1 Greedy repair

The Greedy repair operator uses a slightly altered version of the GRASP heuristic presented in Algorithm 2 to repair the solution. As we start from a partially destroyed solution, the solution set \mathcal{X} is not empty, as is the case in line 6 in Algorithm 2. Further, a construction attempt is started for each of the eight square sequences, where the RCL list length l is set to one. The order of which the square sequences are selected for a construction attempt is random. If a feasible solution is constructed, the search terminates, and the repaired solution is evaluated.

3.4.2 Neighbor repair

This operator is motivated by the expectation that the shifting cost is reduced if vehicles from the same cargo are placed together. Given a partially destroyed solution, the procedure iterates through a list of all empty squares. For each empty square (i, j) , the operator checks if any vehicle in the set of removed vehicles \mathcal{L} could be placed here. If true, the vehicle is placed here if it gets a neighbor, i.e. a vehicle from the same cargo is placed in any of the squares $(i + S_c^L, j)$, $(i - S_c^L, j)$, $(i, j + S_c^W)$, or $(i, j - S_c^W)$. If a square is a neighbor square for more than one vehicle, the largest vehicle is prioritized. When all squares are evaluated, there may still be unplaced vehicles in the list \mathcal{L} . The greedy repair operator is then called to attempt to reinsert these.

3.4.3 Placement repair

It is reasonable to assume that vehicles placed in squares located far from the entry/exit have a lower probability of being shifted. Additionally, we want these squares to be occupied by vehicles that are on the ship for the most sailing legs. For example, it seems sensible to place a vehicle that is loaded in the first port and unloaded in the last at the innermost place on the deck. A list of all empty squares is sorted by the distance to the entry/exit in descending order. Iterating through this list, for each square, all vehicles in the list of removed vehicles \mathcal{L} is attempted reinserted. If possible for more than one vehicle, the vehicle with the highest number of sailing legs ($P_c^U - P_c^L$) is selected.

3.4.4 Random repair

This operator generates a list of all empty squares from the partially destroyed solution. As long as the list is not empty, a square is sampled randomly from the list. For each sampled square, the operator attempts to insert vehicles from the list of removed vehicles \mathcal{L} , prioritizing the largest vehicles. When all empty squares are evaluated or all vehicles are placed on the deck, the operator terminates.

3.5 Shifting cost evaluation

Frequently in optimization problems, calculating a solution's cost $c(\mathcal{X}')$ is trivial. In this case, however, an optimization problem must be solved to provide the value of $c(\mathcal{X}')$, namely the Stowage Plan Evaluation Problem (SPEP). We solve this problem by using the shortest path solution method proposed by Hansen et al. (2017). Given a stowage plan \mathcal{X}' , the method decides which vehicles to shift at each port such that none of the loading/unloading vehicles are blocked, by solving several shortest path problems. Hansen et al. (2017) show that the method is effective in determining the better of two stowage plans, which is what we intend to use the method for in this work, i.e. to compare the shifting cost of the temporary stowage plan $c(\mathcal{X}')$ with the shifting cost of the current one $c(\mathcal{X})$.

Even though the method presented by Hansen et al. (2017) is computationally effective, the time spent evaluating the shifting cost of different solutions make up a substantial amount of the overall runtime of the algorithm. Hence, as mentioned in Sect. 3.1, we try to ensure that only promising solutions are evaluated by implementing two evaluation functions. We use the neighbor function $G(\mathcal{X}')$ and the placement function $P(\mathcal{X}')$ to estimate the quality of a temporary stowage plan \mathcal{X}' , in order to determine whether the shifting cost should be calculated or not. Here, $G(\mathcal{X}')$ gives the number of neighboring vehicles in the solution \mathcal{X}' . A higher number of vehicles from the same cargo placed together on the deck (neighbors) usually has a positive impact on the shifting cost (Hansen et al. 2016). Further, Hansen et al. (2016) show that placing cargoes which are on the ship for the most number of sailing legs farther away from the entry/exit also has a positive impact on the shifting cost, which is estimated by $P(\mathcal{X}')$. For each vehicle, let D_v be the rectilinear distance in squares from its location

(i, j) to the entry/exit (E^I, E^J) multiplied with the number of sailing legs the vehicle is on the deck. Then, $P(\mathcal{X}') = \sum_{v \in \text{all Vehicles}} D_v$, where higher is better.

A solution \mathcal{X}' is regarded as promising if $P(\mathcal{X}')(1+p) \geq P(\mathcal{X})$ or $G(\mathcal{X}')(1+g) \geq G(\mathcal{X})$, where $p \geq 0$ and $g \geq 0$ are the placement and neighbor control parameters, respectively. A solution \mathcal{X}' may be regarded as promising despite being worse than the current solution \mathcal{X} on both quality measures. For example, given $g = 0.1$, $G(\mathcal{X}') = 95$, and $G(\mathcal{X}) = 100$, the logic test returns true despite the temporary solution having less grouped vehicles than the current solution.

3.6 Adaptiveness and acceptance criterion

As for selection of the SPGs, the RWS method is used for selecting the destroy and repair operators in each iteration, where w_{om} is the weight of operator o in segment m . The destroy and repair operators are selected independently. At the start of each segment m , i.e. every M iterations, the scores π_o of all operators o are set to zero. At the end of each iteration, if a new global best solution is found, the scores of the selected destroy operator and repair operator are incremented by the parameter σ_1 . At the end of each segment m , the new weights are calculated as: $w_{o,m+1} = w_{om}(1 - \eta) + \eta\pi_o/\theta_o$, where the reaction factor $\eta \in [0, 1]$ controls how much the current segment scores should influence the weights, while the parameter θ_o counts the number of times operator o was used within the section.

Ropke and Pisinger (2006) state that it seems sensible to accept solutions that are worse than the current solution occasionally, not to get trapped in a local minimum. They propose to use the acceptance criterion from simulated annealing to ensure this. In our case, extensive preliminary testing showed that the heuristic performs better with a simple greedy acceptance criterion than with the proposed simulated annealing acceptance criterion. This may imply that the ALNS heuristic rarely gets stuck in local minima for this stowage problem, and if so, a greedy strategy is sensible. Thus, we accept any solution better than the current solution, which is known as hill-climbing (Burke et al. 2005).

4 Computational study

We have performed a computational study using 512 test instances, generated mainly based on data provided by Wallenius Wilhelmsen Logistics. In Sect. 4.1, the method used to generate the test instances is presented. Next, we present the tuning of the parameters in the ALNS heuristic in Sect. 4.2. In Sect. 4.3, we compare the performance of the ALNS heuristic with solving the MIP using a commercial solver and a stand-alone version of the GRASP construction heuristic. Finally, in Sect. 4.4, we analyze how the solutions are affected by the instance-defining parameters, and the performance of the heuristic's operators is evaluated.

The mathematical model presented in Sect. 3 was implemented in Mosel and solved using XpressMP 1.28.12. The ALNS heuristic was implemented in Java. All compu-

tational experiments have been run on a computer with two Intel Xeon E5-2670v3 2.3 GHz processors, 64 GB of RAM and running on a Linux operating system.

4.1 Test instances

A test instance is defined through six parameters: the deck, # port calls along the voyage, cargo type, # cargoes, filling degree, and grid resolution. Two real decks from two different RoRo ships are used (denoted by A and B), which include information regarding layout, size, unusable space and the location of the entry/exit square. The dimensions of decks A and B are $265 \text{ m} \times 32 \text{ m}$ and $160 \text{ m} \times 32 \text{ m}$, respectively. Four voyages based on real trades with 5, 6, 8, and 10 port calls are used. These are the trades Baltimore–Melbourne, Bremerhaven–Tacoma, Yokohama–Bremerhaven, and Singapore–Baltimore, respectively. The cargoes to be transported are either all standard cars (Car) or a combination of cars and high-and-heavy machinery (HH). Further, each instance consists of either 6, 9, 12, or 15 cargoes.

We define the filling degree as the total area of all vehicles divided by the deck's available area, which is set to either 75% or 90% to represent two realistic levels. The number of vehicles in each cargo is set randomly, while respecting the filling degree and ensuring that each cargo consists of at least one vehicle. Finally, the deck is represented by one of the following four grid resolutions: 100×38 , 200×75 , 300×113 , or 400×150 . The ratio between the number of rows and columns is based on the length and width of a car equivalent unit (CEU), which has an approximate length–width ratio of 4:1.5.

Now, as an example, A-8-HH-12-0.9-200 refers to an instance with deck A and the voyage with eight ports (i.e. the Yokohama–Bremerhaven trade). The cargoes are both cars and high-and-heavy, with a total of 12 cargoes to be handled. The cargoes occupy 90 % of the available area and a grid resolution of 200×75 is used to represent the deck. One instance is created for every unique combination of the parameters, resulting in a total of 512 instances. An asterisks (*) denote all instances within a certain set, such that A-*. *-0.9-* is the set of all instances that use deck A and have a filling degree of 90%. For all test instances, the computing time for the solution methods is limited to 3600 s, and the search is stopped if a solution with zero shifting cost is found before this time. For all instances, we set the cost of shifting a vehicle from cargo c to be $C_c^M = A_c/A^{AVG}$, where A_c is the area of a vehicle from cargo c , and A^{AVG} is the area of the average size vehicle over all cargoes c . Thus, an objective value of two might for example mean that either two average-sized vehicles or one vehicle of twice the average size is shifted during the voyage. The instances are currently available at the web page <http://folk.ntnu.no/joneh>.

4.2 Parameter tuning

Both the GRASP and the ALNS parameters are tuned on 16 representative training instances. We follow the procedure given by Ropke and Pisinger (2006): First, a fair parameter setting is produced by an ad hoc trial-and-error phase. Next, the parameter setting is improved by allowing one parameter to take a number of values, while the rest

Table 1 Parameter tuning results

Parameter	Description	Value
\mathbf{w}^D	Initial square sequence weights	{0.2, 0.1, 0.1, 0.1, 0.1, 0.1, 0.2, 0.1}
\mathbf{w}^L	Initial RCL length weights	{0.4, 0.2, 0.2, 0.1, 0.1}
δ	GRASP score parameter	0.002
ξ	Destruction degree	0.2
σ_1	Score when new global best solution is found	33
η	Reaction factor	0.1
p	Placement parameter	0.2
g	Neighbor parameter	0.2
M	Segment size (number of iterations)	50
w^{dest}	Initial destroy operator weights	{1, 1, 1, 1, 1, 1}
w^{rep}	Initial repair operator weights	{1, 1, 1, 1}

of the parameters are kept fixed. Each instance is solved five times for each setting and the best performing setting is kept. This procedure is repeated for all the parameters included in the test. The settings tested are reported in the following manner: For parameters tuned over an interval, the intervals are shown in square brackets. For parameters tuned over a set of values, curly brackets are used.

For the GRASP construction heuristic, the following parameters are tuned: The square sequence weight parameter \mathbf{w}^D [0, 1], the RCL length weight parameter \mathbf{w}^L [0, 1], the score parameter δ {0.0001, 0.0005, 0.001, 0.002, 0.004}, and the RCL list length l [1, 10] to a maximum of five elements, as exceeding this length limits the heuristic's capabilities. The selected parameter setting is summarized in Table 1.

For the ALNS heuristic, the following parameters are tuned: The destruction degree ξ {0, 0.1, ..., 1.0}, the placement and neighbor parameters p and g {0, 0.05, ..., 0.5}, and the segment size M {25, 50, 100, 150, 200}. The score parameter σ_1 and the reaction factor η are not included in the tuning. These parameters are set to values commonly used in ALNS implementations, see e.g. Ropke and Pisinger (2006) and Gharehgozli et al. (2014). The selected parameter setting is summarized in Table 1.

4.3 Comparing solution methods

In this section, we analyze the computational results of the proposed solution methods. We first present the results of solving the full mathematical model as a MIP using commercial software and comparing it with the ALNS. Then, we compare the ALNS with the GRASP construction heuristic.

To investigate the computational complexity of the problem, we generated a set of artificial small instances. These test instances vary from a grid resolution of 15×15

Table 2 Results of solving the different problem instances both as a MIP and using the ALNS

Instance	MIP			ALNS	
	LB	UB	Time (s)	Obj	Time (s)
6-0.75-15	0	0	6.8	0	1.2
8-0.75-15	0	0	67.6	0	1.1
6-0.90-15	0	0	108.4	0	4.7
8-0.90-15	0	0	86.4	0	1.0
6-0.75-20	0	0	10.1	0	1.0
8-0.75-20	0	0	160.8	0	1.2
6-0.90-20	0	0	73.0	0	1.1
8-0.90-20	0	0	1008.4	0	1.1
6-0.75-25	0	0	51.6	0	1.0
8-0.75-25	0	0	95.8	0	1.1
6-0.90-25	0	0	392.7	0	2.1
8-0.90-25	0	0	271.6	0	1.3
6-0.75-30	0	0	117.2	0	1.0
8-0.75-30	0	0	395.3	0	1.2
6-0.90-30	0	0	3250.8	0	1.2
8-0.90-30	0	39	3600.0	0	1.4
6-0.75-35	0	0	198.8	0	1.0
8-0.75-35	0	0	2805.4	0	1.2
6-0.90-35	0	18	3600.0	0	1.5
8-0.90-35	0	–	3600.0	0	4.5
6-0.75-40	0	0	894.6	0	1.1
8-0.75-40	0	0	1992.2	0	1.4
6-0.90-40	0	–	3600.0	0	1.5
8-0.90-40	0	89	3600.0	0	1.4
6-0.75-45	0	0	2270.3	0	1.3
8-0.75-45	0	11	3600.0	0	1.8
6-0.90-45	0	–	3600.0	0	1.1
8-0.90-45	0	–	3600.0	0	1.5
Average	0	57.375	1537.783	0	1.5

to 45×45 . The filling degree is set to either 75 % or 90 %, and the trade size is either 6 or 8 ports. Here, as an example, 8-0.75-25 refers to an instance with eight port calls, a filling degree of 0.75, and a grid resolution of 25×25 . The instances are solved both as a MIP using a commercial solver and by the ALNS heuristic. The results are summarized in Table 2, where we report the lower bound (LB), upper bound (UB) and computing time (Time) used by the commercial solver, and the objective value (Obj.) and computing time (Time) of the ALNS heuristic. Instances where no upper bound was found within the time limit of 3600 s are marked with an –.

The results in Table 2 clearly show the limitations of solving the MIP model with a commercial solver. For the instances with the smallest grid resolution, i.e. 15×15 ,

Table 3 Shifting cost improvement in percentage of the initial shifting cost for ALNS and GRASP

Set of instances	ALNS		GRASP	
	Time (s)	Improvement (%)	Time (s)	Improvement (%)
*_*_*_*-0.75-100	237.6	99.9	3600	9.1
*_*_*_*-0.75-200	424.7	99.8	3600	7.2
*_*_*_*-0.75-300	778.1	99.5	3600	5.2
*_*_*_*-0.75-400	863.7	99.5	3600	3.2
*_*_*_*-0.9-100	1079.5	99.1	3600	10.1
*_*_*_*-0.9-200	1709.0	96.4	3600	5.4
*_*_*_*-0.9-300	1809.6	96.8	3600	2.6
*_*_*_*-0.9-400	2455.0	95.0	3600	4.4
Average	1169.6	98.2	3600	5.9

A score of 100% indicates a shifting cost of zero is reached
The asterisks (*) denote all instances within a certain set

the average solution time is approximately a minute. With higher grid resolution, the computational times increase rapidly. Similarly, we see that a higher filling degree increases the complexity. Eight of the 28 instances were not solved to optimality within one hour (3600 s). For the ALNS heuristic, all instances are solved to optimality within a few seconds. As the results indicate that the MIP solver can only solve small instances, it is excluded from any further testing.

The GRASP is presented as a heuristic for constructing the initial solution in the ALNS heuristic, where the destroy and repair operators are used in the improvement phase. Alternatively, the GRASP could be used as a stand-alone heuristic where each constructed promising solution is evaluated. Next, we evaluate the two heuristics' ability to improve the initial solution. The results are summarized in Table 3.

For each instance, both heuristics starts from the same initial solution. The time columns in Table 3 show the average computing time for each set of instances. The results show that the GRASP's ability to improve upon the initial solution is limited, with an average improvement of 5.9 %. Further, the GRASP did not find the optimal solution (with zero shifts) to any of the instances. The ALNS found a solution with zero shifts to 381 of the 512 problem instances, with an average improvement of 98.2 %. It is not known whether there exist solutions with zero shifts for the remaining 131 instances. From the results, we see that with a higher filling degree, the average solution times for the ALNS increases. More vehicles on the deck complicates the repair procedures due to fewer placement possibilities. Similarly, higher grid resolutions increase the time spent on destroying, repairing, and evaluating solutions, which also affect the computational times.

It is clear that using the ALNS to improve the initial solution yields significantly better results than iteratively creating stowage plans from scratch using the GRASP. Further, we conclude that time is better spent on improving the initial solution using the ALNS than searching for a better initial solution using the GRASP. Still, the GRASP constructed at least one feasible solution to all instances, which is the primary purpose

of the procedure in the proposed ALNS framework. In all further analyses, only the ALNS heuristic is used.

4.4 Extensive testing of the ALNS heuristic

In this section, we analyze the computational results from the ALNS heuristic further. We first discuss which instance-defining parameters that influence shifting cost the most. Next, the performances of the operators of the ALNS heuristic are analyzed, and the shifting cost evaluation times are reported. Finally, we illustrate a solution to the problem.

The computational results for various sets of instances are summarized in Table 4. We see that the average objective value is 1.09, where an objective value of one means that one average-sized vehicle is shifted during the voyage. Further, we see that the heuristic succeeded in creating a stowage plan with zero shifts for 74.41% of the instances. On average, 0.14 vehicles are shifted per port call. We argue that this is acceptable for any practical use, which shows the potential for planners to use the stowage heuristic in daily operations. We discuss practical use more in detail at the end of this section.

Two different deck layouts are used in this study. The results show no clear difference in the objective value, whether deck A or B is used. As the decks on a RoRo ship usually are large open areas, the deck layout will most likely not have a considerable impact on the shifting cost. However, for especially small decks with several obstacles, the shifting cost will most likely be higher due to the layout. Further, we see that with more port calls, the average objective value increases. More port pairs complicate the procedure of placing vehicles from different cargoes on the deck in a smart way to avoid shift.

We also see from Table 4 that the set of instances containing high-and-heavy vehicles have more shifts on average than the instances with only cars. This is probably because larger vehicles need more space to be loaded and unloaded. While the size of the vehicles matter, the number of cargoes seems to have a negligible impact on the shifting cost of a solution. We know that if only one cargo is transported, the shifting cost is zero, as all vehicles are loaded and unloaded at the same port. Similarly, if there are ten cargoes that are all loaded and unloaded at the same port, the shifting cost is still zero. Hence, the number of cargoes do not directly affect the shifting cost. However, we know that a higher number of unique port pairs impact the shifting cost negatively. Thus, as more cargoes increase the probability of having transportation between more unique port pairs, it indirectly influences the shifting cost.

From the results, we see that filling degree has the highest impact on the shifting cost of a stowage plan. The average shifting costs are 0.22 and 1.96 for 75% and 90% filling degree, respectively. This is simply due to more vehicles on the deck that potentially could induce shifting, in addition to less free space on the deck to use for loading and unloading vehicles. Finally, it is clear that a higher grid resolution increases the objective value, which may seem counter-intuitive. This is because higher grid resolutions increase the shifting cost evaluation times, which we discuss in more details later. When fewer iterations are achieved within the computing time limit, the

Table 4 Computational results for various sets of instances solved with the ALNS heuristic

Set of instances	Obj	Time	# Shifts per port	Zero shifts %
A-**-**	1.17	1320.2	0.14	71.48
B-**-**	1.02	1019.1	0.13	77.34
*-5-**-**	0.26	692.3	0.05	86.72
*-6-**-**	0.59	898.9	0.10	82.81
*-8-**-**	1.71	1532.0	0.21	63.28
*-10-**-**	1.81	1555.3	0.18	64.84
--Car-**-**	0.59	1004.2	0.07	81.25
--HH-**-**	1.59	1335.1	0.20	67.58
--6-**-**	1.00	844.0	0.12	78.91
--9-**-**	0.94	910.1	0.12	80.47
--12-**-**	1.30	1450.9	0.17	71.09
--15-**-**	1.12	1473.5	0.14	67.19
--*-0.75-**-**	0.22	576.0	0.03	90.23
--*-0.9-**-**	1.96	1763.3	0.24	58.59
--*-100-**-**	0.27	658.6	0.03	84.38
--*-200-**-**	1.10	1066.9	0.13	75.00
--*-300-**-**	1.20	1293.8	0.14	72.66
--*-400-**-**	1.79	1659.3	0.23	65.63
Average	1.09	1169.6	0.14	74.41

The asterisks (*) denote all instances within a certain set

solution space is less explored, which reduces the probability of finding an optimal solution.

Performance of the operators of the ALNS heuristic

The majority of the destroy and repair operators are developed specifically for the 2DRSSP, hence they are new to the literature. Table 5 summarizes each operator's performance for various sets of instances. Each operator's performance is given by the number of times it is used to reduce the shifting cost, divided by the total number of improvements. For both the destroy and repair operators, the results show that the cargo type and filling degree parameters impact the operators' performance the most.

From the results, we see that the Neighbor operator, which removes vehicles without a neighbor, scores best on average. This operator performs especially well for instances with cars only, with scores of 28.0% and 23.2% for instances with low (75 %) and high (90 %) filling degrees, respectively. However, for the set of instances with a high filling degree and high-and-heavy (HH) cargoes, the score drops to 10.7%. A reason for this may be that removing vehicles without a neighbor may give a partially destroyed stowage plan with many holes. Repairing such a stowage plan can be hard to accomplish when the filling degree is high. For example, the operator removes a large high-and-heavy vehicle from the stowage plan, hence, creating a hole in the

Table 5 The destroy operators' performance for different sets of instances

Set of instances	Destroy operators (%)					
	Neighbor	Area	Random	Port	Shifting cost	Route
_-Car-*-0.75-*	28.0	10.8	5.3	18.0	18.9	18.9
_-HH-*-0.75-*	24.7	11.5	6.8	19.5	18.7	18.8
_-Car-*-0.9-*	23.2	14.3	2.9	17.8	21.0	20.8
_-HH-*-0.9-*	10.7	12.3	4.1	25.3	24.2	23.3
Average	21.6	12.2	4.8	20.2	20.7	20.5

The asterisks (*) denote all instances within a certain set

Table 6 The repair operators' performance for different sets of instances

Set of instances	Repair operators (%)			
	Greedy	Neighbor	Placement	Random
_-Car-*-0.75-*	17.0	52.3	18.9	11.8
_-HH-*-0.75-*	16.5	55.0	21.0	7.4
_-Car-*-0.9-*	27.6	43.9	26.4	2.1
_-HH-*-0.9-*	39.4	37.8	22.1	0.7
Average	25.1	47.3	22.1	5.5

The asterisks (*) denote all instances within a certain set

stowage plan. If a small vehicle is reinserted in this space, the remaining area around the small vehicle becomes unusable. With less available usable area, feasibility could now become a challenge. For this difficult set of instances, *_*-HH-*-0.9-*, the Port, Shifting cost, and Route operators perform better. Removing vehicles that are to be loaded or unloaded at the same port yield promising results for this specific set of instances, with a score of 25.3%. Both the Shifting cost and Route operators use information from the stowage plan evaluation problem to select which vehicles to remove, either by removing the vehicles that block other loading/unloading vehicles the most, or by removing the vehicles that require most shifts to be loaded or unloaded. The results show that these operators perform well on all sets of instances.

Table 6 shows the repair operators' performance. It is clear that placing vehicles from the same cargo together has its merits, as the Neighbor repair operator has a score of 47.3%. By placing vehicles from the same cargo together, the area is most often better utilized. Additionally, it has a positive impact on the shifting cost. If a vehicle can be unloaded without inducing shifting, then so will a neighboring vehicle by following the same exit route. Though, we see that with increasing filling degree, the Neighbor operator's score is reduced. With limited space, forcing vehicles into neighbor positions may create unused areas around smaller vehicles, as was also the case for the Neighbor destroy operator. The Placement repair operator seems to perform equally well for different sets of instances. The Greedy repair operates without any specific logic by inserting the largest vehicles first. For more difficult instances, this increases the probability of feasibility. Thus, by succeeding in repairing solutions

Table 7 The average shifting cost evaluation time for various sets of instances for different grid resolutions

Set of instances	Grid resolution			
	100	200	300	400
-5-6*-c	0.02	0.04	0.11	0.22
-6-6*-c	0.01	0.04	0.10	0.20
-8-6*-c	0.01	0.04	0.11	0.25
-10-6*-c	0.01	0.05	0.11	0.25
-5-9*-c	0.03	0.08	0.18	0.42
-6-9*-c	0.02	0.07	0.15	0.39
-8-9*-c	0.02	0.09	0.16	0.27
-10-9*-c	0.02	0.06	0.13	0.35
-5-12*-c	0.02	0.07	0.18	0.42
-6-12*-c	0.01	0.07	0.16	0.67
-8-12*-c	0.02	0.06	0.17	0.67
-10-12*-c	0.01	0.07	0.19	0.59
-5-15*-c	0.03	0.09	0.19	0.88
-6-15*-c	0.02	0.07	0.32	1.03
-8-15*-c	0.02	0.08	0.19	0.94
-10-15*-c	0.01	0.08	0.30	1.07
Average	0.02	0.07	0.17	0.54

All results are in seconds per iteration

The asterisks (*) denote all instances within a certain set c indicates that parameter is given by the column header

more often than the other repair operators, the probability of generating stowage plans with lower shifting cost increases. As for the Random destroy operator, the Random repair operator scores poorly, especially for instances with a high filling degree.

Shifting cost evaluation time

The majority of the time in each iteration is spent on calculating the shifting cost of the repaired stowage plans (i.e. solving the SPEP). Here, we examine how the different parameters that define an instance influence the evaluation times. The results are summarized in Table 7.

We see that the average time used to solve the SPEP ranges from 0.02 s per iteration for the lowest grid resolution to 0.54 s per iteration for the highest grid resolution. This parameter has a high impact on the solution times of the SPEP, as most of the variables in the problem have a row and a column index. Further, we see that with more cargoes, the solution times increase, even though the total number of vehicles placed on the deck is approximately the same for the different sets of instances. The shortest path solution method used to solve the SPEP solves two SPPs per cargo, one for the cargo's loading and unloading port, respectively. While the number of vehicles in each cargo influences the solution times due to more target nodes in the network, i.e. one target node for each vehicle, the number of cargoes greatly outweighs this

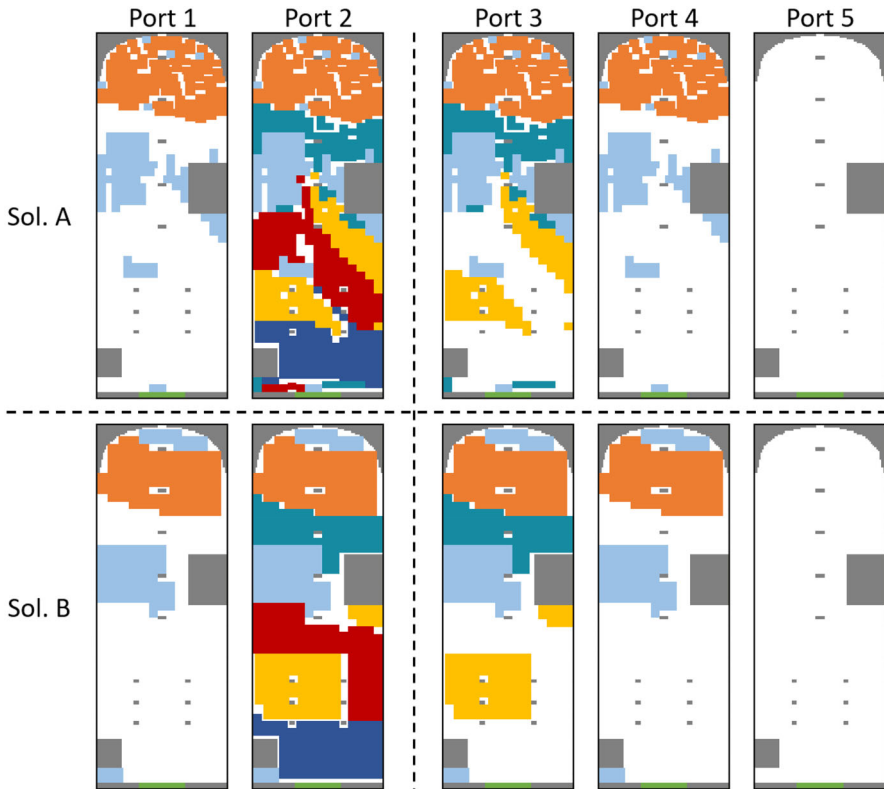


Fig. 4 Two solutions with zero shifts to the problem instance A-5-Car-6-0.9-200 is shown. The gray color gives the unusable spaces, while the remaining six different colors represent vehicles from each of the six cargoes placed on the deck

effect. Finally, we see that the number of ports does not seem to affect the evaluation times significantly. While not reported in Table 7, the solution times seem not to be considerably influenced by any of the other instance-defining parameters, namely deck layout, cargo type, and filling degree.

Solving practical problems

In Fig. 4, two solutions to the problem instance A-5-Car-6-0.9-200 is illustrated. In this problem instance, vehicles are loaded at ports 1 and 2, and unloaded at ports 3-5. The entry/exit area is located at the stern (i.e. back of the ship), which is marked in green in Fig. 4. The gray areas are unusable spaces. Each cargo consists of 83 to 112 vehicles and is represented by a unique color. A total of 585 vehicles are placed on the deck after loading at port 2. At each port, the stowage plan after loading or unloading is shown. As seen from the figure, all cargoes are onboard the ship after the second port call.

Solution A is the solution found by the ALNS heuristic during computational testing and does not require any shifting. Looking at the solution, we see that despite being optimal with respect to the mathematical formulation, the solution is complicated to implement in a practical setting. For example, we see that the vehicles marked in orange are scattered in the upper part of the deck.

Solution B is also constructed using the ALNS heuristic. However, a minor alteration to the stopping criteria is made. Instead of stopping the search when a solution with zero shifts is found (or after 3600 s), we let the heuristic search for an additional 30 s. During this search, we accept any solution with a higher number of neighboring vehicles than the current solution, given that the shifting cost remains unchanged. We see from the figure that both solutions have the cargoes placed roughly in the same area. However, increasing the number of neighboring vehicles has resulted in a more structured and visually pleasing stowage plan than Solution A. From a planners perspective, it is clear that Solution B is the preferred stowage plan as it is easy to replicate in a practical setting. Further, from a practical viewpoint, it may also be desirable to load/unload all vehicles in one cargo at a time. This is more likely to be possible with a structured stowage plan, as the probability of having a random vehicle placed within a group of vehicles from the same cargo is lower. For example, we see from Solution A that some light blue vehicles are placed within the group of vehicles from the orange cargo. While it is shown that the ALNS heuristic is capable of generating stowage plans with low shifting cost as well as structured solutions, the trade-off between shifting costs and logically structured stowage plans is not addressed in this work and is left as a promising venue for further research.

5 Concluding remarks

We have considered the two-dimensional RoRo ship stowage problem for one deck (2DRSSP), which is a generalized version of the planar storage location assignment problem. The 2DRSSP considers a ship sailing along a predefined voyage, visiting a set of ports. At each port, cargoes are to be loaded or unloaded. In order to keep the time spent on loading and unloading vehicles to a minimum and, hence, reduce the time spent in port, a good stowage plan should minimize unnecessary movement of vehicles. This undesirable action of moving vehicles to enable loading or unloading of other vehicles is referred to as shifting. In previous studies, this shifting aspect has either been oversimplified or handled implicitly using heuristic placement strategies. In this work, we stress the importance of handling shifting in such a manner that the resulting stowage plans are of practical use.

We have proposed a novel mixed integer programming model for the 2DRSSP. A new way of modeling the shifting aspect is a central part of the model. We have also proposed an adaptive large neighborhood search (ALNS) heuristic for solving realistic problem instances. The computational results showed that the resulting stowage plans are of practical use, with an average of approximately one shift per instance. While having its merits as is, the solution method could also be used as a building block in future work, extending the problem to cover multiple decks and stability restrictions. Stability calculations are complex issues that are usually handled by naval architects.

However, given a procedure for checking the stability for a given stowage arrangement, it would be fairly easy to incorporate that into our ALNS heuristic by using this procedure to check stability whenever the heuristic finds a new best solution. This would also make sure that we get a solution that also respects the stability restrictions.

Acknowledgements Open Access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital). We are grateful to Ivar Hukkelberg who provided input at an early stage of this work. We also acknowledge the comments and suggestions provided by two anonymous reviewers which helped us improve the paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Bayliss, C., Currie, C.S., Bennell, J.A., Martinez-Sykora, A.: Dynamic pricing for vehicle ferries: using packing and simulation to optimize revenues. *Eur. J. Oper. Res.* **273**, 288–304 (2019)
- Burke, E.K., Kendall, G., et al.: *Search Methodologies*. Springer, Berlin (2005)
- Chen, P., Fu, Z., Lim, A., Rodrigues, B.: Two-dimensional packing for irregular shaped objects. In: *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2003. IEEE (2003)
- Cordeau, J.F., Laporte, G., Moccia, L., Sorrentino, G.: Optimizing yard assignment in an automotive transshipment terminal. *Eur. J. Oper. Res.* **215**, 149–160 (2011)
- Ding, D., Chou, M.C.: Stowage planning for container ships: a heuristic algorithm to reduce the number of shifts. *Eur. J. Oper. Res.* **246**, 242–249 (2015)
- Feo, T.A., Resende, M.G.: A probabilistic heuristic for a computationally difficult set covering problem. *Oper. Res. Lett.* **8**, 67–71 (1989)
- Feo, T.A., Resende, M.G.: Greedy randomized adaptive search procedures. *J. Global Optim.* **6**, 109–133 (1995)
- Gharehgozli, A.H., Laporte, G., Yu, Y., de Koster, R.: Scheduling twin yard cranes in a container block. *Transp. Sci.* **49**, 686–705 (2014)
- Gu, J., Goetschalckx, M., McGinnis, L.F.: Research on warehouse operation: a comprehensive review. *Eur. J. Oper. Res.* **177**, 1–21 (2007)
- Haims, M.J., Freeman, H.: A multistage solution of the template-layout problem. *IEEE Trans. Syst. Sci. Cybern.* **6**, 145–151 (1970)
- Hansen, J.R., Fagerholt, K., Stålhane, M.: A shortest path heuristic for evaluating the quality of stowage plans in roll-on roll-off liner shipping. In: *Lecture Notes in Computer Science*, vol. 10572, pp. 351–365 (2017)
- Hansen, J.R., Hukkelberg, I., Fagerholt, K., Stålhane, M., Rakke, J.G.: 2d-packing with an application to stowage in roll-on roll-off liner shipping. In: *Lecture Notes in Computer Science*, vol. 9855, pp. 35–49 (2016)
- Hemmelmayr, V.C., Cordeau, J.F., Crainic, T.G.: An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Comput. Oper. Res.* **39**, 3215–3228 (2012)
- Iris, C., Pacino, D.: A survey on the ship loading problem. In: *Lecture Notes in Computer Science*, vol. 9335, pp. 238–251 (2015)
- Iris, C., Pacino, D., Ropke, S.: Improved formulations and an Adaptive Large Neighborhood Search heuristic for the integrated berth allocation and quay crane assignment problem. *Transp. Res. Part E Logist. Transp. Rev.* **105**, 123–147 (2017)

- Iris, C., Christensen, J., Pacino, D., Ropke, S.: Flexible ship loading problem with transfer vehicle assignment and scheduling. *Transp. Res. Part B* **111**, 39–56 (2018)
- Mattfeld, D.C., Orth, H.: The allocation of storage space for transshipment in vehicle distribution. *OR Spectrum* **28**, 681–703 (2006)
- Mauri, G.R., Ribeiro, G.M., Lorena, L.A.N., Laporte, G.: An adaptive large neighborhood search for the discrete and continuous Berth allocation problem. *Comput. Oper. Res.* **70**, 140–154 (2016)
- Monaco, M.F., Sammarra, M., Sorrentino, G.: The terminal-oriented ship stowage planning problem. *Eur. J. Oper. Res.* **239**, 256–265 (2014)
- Øvstebø, B.O., Hvattum, L.M., Fagerholt, K.: Optimization of stowage plans for ro-ro ships. *Comput. Oper. Res.* **38**, 1425–1434 (2011a)
- Øvstebø, B.O., Hvattum, L.M., Fagerholt, K.: Routing and scheduling of ro-ro ships with stowage constraints. *Transp. Res. Part C Emerg. Technol.* **19**, 1225–1242 (2011b)
- Park, C., Seo, J.: Mathematical modeling and solving procedure of the planar storage location assignment problem. *Comput. Ind. Eng.* **57**, 1062–1071 (2009)
- Park, C., Seo, J.: Comparing heuristic algorithms of the planar storage location assignment problem. *Transp. Res. Part E Logist. Transp. Rev.* **46**, 171–185 (2010)
- Prais, M., Ribeiro, C.C.: Parameter variation in grasp procedures. *Investig. Oper.* **9**, 1–20 (2000)
- Puisa, R.: Optimal stowage on ro-ro decks for efficiency and safety. *J. Mar. Eng. Technol.* (2018). <https://doi.org/10.1080/20464177.2018.1516942>
- Resende, M.G., Ribeiro, C.C.: Grasp: greedy randomized adaptive search procedures. In: Burke, E., Kendall, G. (eds.) *Search Methodologies*, pp. 287–312. Springer, Boston (2014)
- Reyes, J., Solano-Charris, E., Montoya-Torres, J.: The storage location assignment problem: a literature review. *Int. J. Ind. Eng. Comput.* **10**, 199–224 (2019)
- Ribeiro, G.M., Laporte, G.: An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Comput. Oper. Res.* **39**, 728–735 (2012)
- Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **40**, 455–472 (2006)
- Seixas, M.P., Mendes, A.B., Pereira Barretto, M.R., Da Cunha, C.B., Brinati, M.A., Cruz, R.E., Wu, Y., Wilson, P.A.: A heuristic approach to stowing general cargo into platform supply vessels. *J. Oper. Res. Soc.* **67**, 148–158 (2016)
- Tao, N., Jiang, Z., Qu, S.: Assembly block location and sequencing for flat transporters in a planar storage yard of shipyards. *Int. J. Prod. Res.* **51**, 4289–4301 (2013)
- Tierney, K., Pacino, D., Jensen, R.M.: On the complexity of container stowage planning problems. *Discrete Appl. Math.* **169**, 225–230 (2014)
- Wang, P.: Two algorithms for constrained two-dimensional cutting stock problems. *Oper. Res.* **31**, 573–586 (1983)
- Wei-Ying, Z., Yan, L., Zhuo-Shang, J.: Model and algorithm for container ship stowage planning based on bin-packing problem. *J. Mar. Sci. Appl.* **4**, 30–36 (2005)
- Xiao, J., Zheng, L.: A correlated storage location assignment problem in a single-block-multi-aisles warehouse considering bom information. *Int. J. Prod. Res.* **48**, 1321–1338 (2010)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.