On the Complexity of Slot-Based RoRo Stowage Planning

Gerth Stølting Brodal[1][0000−0001−9054−915X]*, Martin Olsen[3][0000−0002−2740−545X], Dario Pacino[2][0000−0002−7255−004X], and Oliver Rise Thomsen[2][0009−0005−4758−6410]

[1] Department of Computer Science, Aarhus University, Denmark, gerth@cs.au.dk
[2] Department of Technology, Management and Economics, Technical University of Denmark, {darpa,olitho}@dtu.dk
[3] Department of Business Development and Technology, Aarhus University, Denmark, martino@btech.au.dk

Abstract. We consider the computational complexity of stowage planning with a specific view of Roll-on/Roll-off (RoRo) vessels. Stowage planning is the process of determining where cargo is loaded on a vessel. A key part of a stowage plan is to avoid shifts of cargo, i.e., unnecessary cargo movements. We focus on the slot-based RoRo variant where the deck of a vessel has a predefined set of equally sized positions and each position can be occupied by a cargo item. We show that it is $\mathcal{NP}$-hard to decide if there is a slot-based RoRo plan with no shifts. We also show how to generate slot-based RoRo instances requiring shifts. Finally, we present a greedy linear time algorithm always generating a plan with no shifts for a special RoRo case.

Keywords: Stowage planning · RoRo shipping · Computational complexity

1 Introduction

In shipping operations a central operational component is stowage planning. A stowage plan determines the placement of cargo on a vessel, and depending on its application also how the cargo is to be secured and loaded. A feasible stowage plan must adhere to a number of restrictions to ensure the seaworthiness of the vessel – for example that dangerous cargo is securely positioned. An optimal stowage plan can have different objectives depending on the specific industrial application, e.g., minimizing time at port, reducing costs, maximizing revenue, etc. Examples of industrial application of stowage planning can be found in container shipping, Roll-on Roll-off (RoRo) shipping, and bulk shipping.

Container vessels are stowed from above using specialized cranes. The containers are loaded in stacks and the stow area is divided into rows and bays where containers are placed [14]. Containers are standardized; most containers are either 20 or 40 foot. Container vessels usually operate on a fixed schedule and visit

* Supported by Independent Research Fund Denmark, grant 9131-00113B.

multiple ports during a voyage. Multi-port voyages have multiple Port Of Load (POL) and Port Of Discharge (POD). In multi-port voyages, *restows* can occur. A restow is when a cargo item must be moved on the vessel before its POD and it leads to higher turn-around times due to unnecessary crane moves. Restows are often also referred to as *shifts* or *involuntary moves*. For the remainder of this paper, we will refer to these additional cargo movements as *shifts*.

Bulk carriers are a versatile type of vessel. They store their cargo in a set of tanks, where typical products are usually bulk like coal, grain or ore, but can also be finished products like steel coils or cement elements. Depending on the cargo type, the loading is done with cranes or conveyor belts from above. Bulk carriers can carry different products in each tank on a voyage, but they must ensure that the tanks are suitable for the specific product and compatible with the other products being carried [15]. For finished products like steel coils, the stow plan must also include how to stack them properly [11].

RoRo vessels carry any wheeled cargo and are thus flexible. The loading of RoRo vessels differs from containers and bulk since the cargo is rolled onto the deck through ramps. The cargo can be of any size but usually consists of trailers or cars. RoRo vessels are both used for short and ocean-sea shipping. In short-sea shipping (e.g., within Europe), they sail on services between two ports, thus having a single POL and POD [8]. In ocean-sea shipping, they visit multiple ports, increasing the complexity of the stow plan [16] due to the possible occurrence of shifts.

(a) (b)

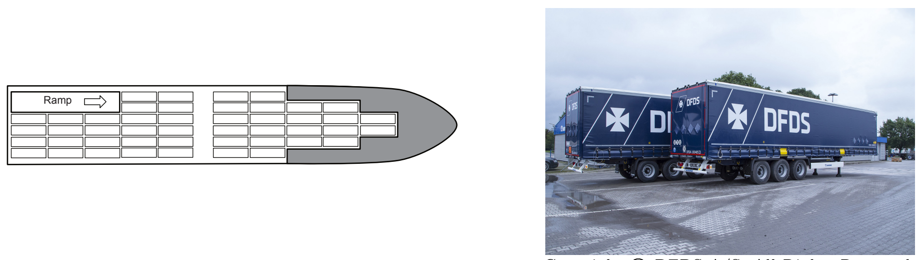


Fig. 1. (a) Slots in deck layout. (b) Standard cargo trailer.

RoRo vessels are composed of several decks, which can be accessed through internal ramps. As the decks provide free positioning of cargo, what distinguishes academic studies is the way cargo positions are modelled. The slot-based RoRo stowage planning problem considers the decks to have a predefined set of equally sized positions, and hence assumes homogenous cargo. Though this is a restrictive constraint for ocean-going vessels, it is surprisingly relevant for short-sea shipping where the majority of cargo is standard sized trailers and special platforms (Mafi RoRo trailers). Fig. 1(a) shows an example slot arrangement on a deck, and Fig. 1(b) a typical cargo trailer. Though the layout of the slots resembles that of a container vessel, RoRo vessel cargo can be moved freely within the deck and is not restricted by stacks. As for any stowage planning problems, vessel stability, stress forces, and the use of ballast water are also relevant parts of the problem [8].

With this paper we aim at contributing to the theory of stowage planning by studying the computational complexity of RoRo stowage planning. In particular, we study the slot-based stowage planning problem of a single deck for a vessel that visits multiple ports. Our contributions are three-fold. First, we prove that the slot-based RoRo stowage planning problem is $\mathcal{NP}$-complete for a stylized vessel definition, which is then extended to a more general vessel layout. Second, we show how our result can be used to generate computationally hard instances for the problem. Existing benchmarks are mainly based on instances requiring zero shifts, hence existing solution approaches are not well tested on harder instances. Third, we present a greedy linear time algorithm always producing a plan with no shifts for a special case of the RoRo stowage planning problem.

This paper is organised as follows. Section 2 presents relevant computational complexity results on stowage planning problems. In Section 3 we prove that the slot-based RoRo stowage planning is $\mathcal{NP}$-complete on a symmetric deck layout, an assumption that is then lifted in Section 4. Section 5 shows how insights from the complexity proof can be leveraged to create meaningful benchmark instances. Section 6 presents the greedy algorithm. Finally, Section 7 concludes the paper.

2 Related work

The following overview of computational complexity results focuses on stowage problems related to the shipping industry. In container stowage, the planning problem is represented as a stacking problem, where containers are assigned to vertical LIFO (last-in, first-out) stacks in bays of the vessel's cargo hold. The goal is to stack the containers to minimize the number of shifts needed during a voyage. The first published complexity result [1] shows that, given a number of stacks with infinite capacity, it is $\mathcal{NP}$-complete to determine if there is a plan for stowing containers with no shifts. The proof is based on a reduction from the problem of coloring intervals such that overlapping[4] intervals do not get the same color [4]. The stacking problem is also $\mathcal{NP}$-hard for any fixed bound $h \geq 6$ on the stacking height [3]. This can be shown by a reduction from the coloring problem on permutation graphs [7].

If the number of stacks and the bound of the stacking height are fixed constants, then there is a polynomial time algorithm for deciding if the stacking problem can be solved using no more than a given number of shifts as shown in [12]. Furthermore, [12] introduces the hatch overstow problem, where containers are placed on top of the hatches on the container vessel and the objective

[4] Two intervals overlap if they intersect and none of them are contained in the other interval.

is to minimize the number of times the hatches are accessed. The hatch overstow problem is then shown to be $\mathcal{NP}$-complete using a reduction from the set covering problem.

Stowage planning of bulk carriers is about the allocation of products to the tanks for the carrier. The tank allocation problem (TAP) is $\mathcal{NP}$-complete [6] and has shown to be computationally intractable utilizing a reduction from the partition problem.

Differently than in container and bulk vessels, where cargo is restricted to specific areas (stacks or tanks), RoRo vessels have an open deck layout, where wheeled cargo is loaded through ramps. The computational complexity of this problem depends heavily on how the layout of the vessel is represented. In the literature, four main methods have been proposed: lane-based, grid-based, slot-based, and the hybrid-slot approach. The RoRo stowage planning problem (RSSP) introduced in [16] models the deck layout with lanes. The lane representation partitions the deck space into a set of lanes where cargo is assigned, thus mirroring the real world. The grid representation allows the free positioning of cargo on the deck. To identify the position of cargo, a coordinate system in the form of a grid of points is used [5]. The slot and hybrid-slot representations create a set of predefined slots on the deck area and then assign cargo to them [8,10].

In terms of computational complexity, the RSSP variant from [16] is $\mathcal{NP}$-hard which is proved by the authors by reduction from the knapsack problem using heterogeneous cargo items. Though no computational complexity results have been published for the grid representation, it would be trivial to derive a reduction from 2D-packing. No results have been published for the slot and hybrid-slot representations.

In the case of RoRo stowage planning, we see a gap in the understanding of the computational complexity of planning using the slot and hybrid-slot representation. The slot representation is the focus of this paper.

3 Slot-based RoRo stowage planning is $\mathcal{NP}$-complete

We now turn our attention to a simple version of the slot-based RoRo problem described in the introduction. The deck layouts we require are depicted in Fig. 2(a) with W and L denoting the width and length of the deck, respectively. In Section 4 we generalize the reduction to apply to other deck layouts. The deck layout in Fig. 2(a) has a total capacity of $N = LW - \left\lfloor \frac{W-1}{2} \right\rfloor \left(1 + \left\lfloor \frac{W}{2} \right\rfloor\right)$ cargos. There will be two center lanes with length L if W is even. The cargo can enter and leave using W ramps at the stern of the vessel. One cargo item can occupy one slot, and cargo can only move from a slot to one of the (up to four) neighbouring slots sharing an edge with the slot.

The RoRo problem asks whether it is possible to load and unload the cargo items with no *shifts*, where a shift as described earlier is a move of a cargo item not used for loading or unloading the item at a port. We now formally define the RoRo problem and name it RoRo-Slot-No-Shift.

Definition 1. *The* RoRo-Slot-No-Shift *problem is defined as follows:*

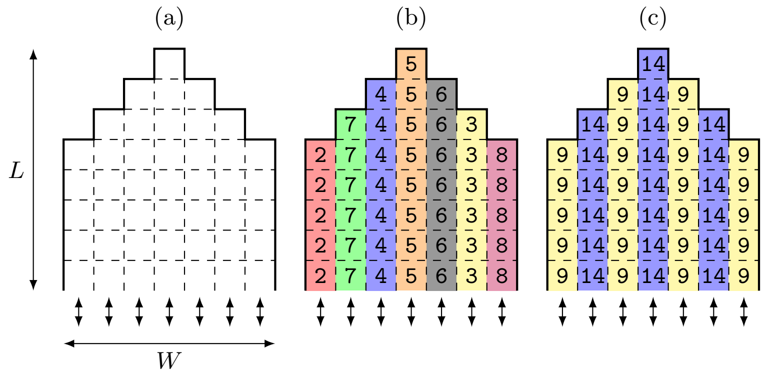


Fig. 2. (a) The layout of the deck of a vessel for $W = 7$ and $L = 8$. Cargo can enter and leave the vessel using W ramps at the stern. (b) The deck configuration right after Step 1, where the vessel has left Port 1. The numbers and colors in the slots represent the destination ports of the cargo. For every destination port, the cargo is placed in a single lane. (c) The deck configuration when Step 2 has finished and the vessel has left Port 8. For the Ports 9 to 13, the deck will act like four uncapacitated LIFO columns because the cargo has to enter and leave using a single lane only ($C = 4$, $p = 5$ and $n = 5$).

– *Input: A $P \times P$ matrix $R = \{r_{i,j}\}$ and numbers W and L, where P is the number of ports. The entry $r_{i,j}$ in R is the number of cargo items that has to be transported from Port i to Port j, and the numbers W and L represent the width and length of the deck, respectively.*
– *Question: Is there a plan for transporting the cargo items with no shifts?*

We now revisit the stacking problem considered by Avriel et al. [1], that was introduced in Section 2, where we have to decide if it is possible to transport some containers with no shifts using a given number of stacks with infinite capacity. We will refer to this problem as the Stacking-No-Shift problem. We remind the reader that Avriel et al. have shown that the Stacking-No-Shift problem is $\mathcal{NP}$-complete. As demonstrated by Avriel et al., we can replace a group of containers with the same origin and destination with a single container when we consider the Stacking-No-Shift problem using stacks with infinite capacity. This is the reason why a matrix with binary entries is used in the following formal definition of the Stacking-No-Shift problem. We assume that a vessel visits Ports 1 to p in increasing order.

Definition 2. *The* Stacking-No-Shift *problem is defined as follows:*

– *Input: A $p \times p$ matrix $S = \{s_{i,j}\}$ and a number C, where p is the number of ports. The entry $s_{i,j}$ in S is 1 if there are containers to be transported from Port i to Port j and 0 otherwise, and the number C indicates the number of infinite capacity vertical stacks to be used for stowing the containers.*
– *Question: Does a plan exist with no shifts for transporting the containers?*

In the following we show that the RoRo-Slot-No-Shift problem is $\mathcal{NP}$-complete by using reduction from the Stacking-No-Shift problem. The intuition of the proof is to reduce a Stacking-No-Shift problem to a RoRo-Slot-No-Shift problem by transforming the deck of the vessel in the RoRo-Slot-No-Shift problem into a set of LIFO columns. The key trick is to force every second lane of the deck to be filled with cargo that has to go to the final port and leave all the other lanes empty. After this has been done, we represent a container from the Stacking-No-Shift problem by a cargo item in the RoRo-Slot-No-Shift problem with a corresponding origin and destination. In this way, we can solve the Stacking-No-Shift problem efficiently if we have access to an efficient algorithm for the RoRo-Slot-No-Shift problem. To put it short, we reduce a stacking problem to a RoRo problem by using the lanes of a RoRo deck to simulate the stacks from the stacking problem.

We are considering a restricted version of the problem where the deck is not too big compared to the size of the input. This makes it possible to prove that the problem is in $\mathcal{NP}$ since an instance with a positive answer to the question (a yes-instance) can be verified in polynomial time by simulating a plan for loading and unloading the cargo. If the deck was too big, this simulation could potentially take super-polynomial time in the size of the input. It should be noted that a corollary of Theorem 1 is that the unrestricted RoRo-Slot-No-Shift problem is $\mathcal{NP}$-hard. The formal details will now follow.

Theorem 1. *The* RoRo-Slot-No-Shift *problem is $\mathcal{NP}$-complete restricted to instances with $WL \leq P^3$.*

Proof. If there is a plan with no shifts, then there will be no more than PWL cargo involved, and the specific route on the deck for each cargo contains no more than WL steps. The restriction $WL \leq P^3$ implies that the RoRo-Slot-No-Shift problem is in $\mathcal{NP}$ since a yes-instance can be verified in polynomial time by simulating a plan with no shifts.

We now show how to transform a Stacking-No-Shift instance into a RoRo-Slot-No-Shift instance with identical answers to the questions posed in the definitions of the problems. First, we provide the details for the case that C is even. At the end of the proof, the case of an odd value for C is addressed.

A Stacking-No-Shift instance (S, C), where S is a $p \times p$ 0-1-matrix and C the number of stacks, is transformed into a RoRo-Slot-No-Shift instance (R, W, L) with $P = 2C + p + 1$, $W = 2C - 1$ and $L = n + C - 1$, where $n = \sum s_{i,j}$ is the number of 1-entries in S. Please note that the RoRo-Slot-No-Shift instance can be constructed in polynomial time since we can assume $C < p$ (otherwise the Stacking-No-Shift problem is trivially solvable by having one stack per destination port, and we can transform the Stacking-No-Shift instance into a trivial yes-instance of the RoRo-Slot-No-Shift problem). Please also note that $WL = (2C - 1)(n + C - 1) \leq P(p^2 + C - 1) \leq P(p + C - 1)^2 \leq P^3$ since $n \leq p^2$.

The matrix R is defined in three steps in order to make the proof easier to follow. In the first step, the deck is filled with cargo at Port 1. All the cargo has to go to the Ports 2, 3, ..., $2C$. The details for Step 1 are as follows:

$$\begin{array}{ccccc} \begin{array}{r} r_{1,2} = n \\ r_{1,3} = n+1 \end{array} & \cdots & \begin{array}{r} r_{1,C} = n+C-2 \\ r_{1,C+1} = n+C-1 \\ r_{1,C+2} = n+C-2 \end{array} & \cdots & \begin{array}{r} r_{1,2C-1} = n+1 \\ r_{1,2C} = n \end{array} \end{array}$$

In Step 2, the vessel unloads the cargo from Step 1 in $W = 2C - 1$ ports, Ports 2, 3, ..., $2C$. For each of these ports, cargo is loaded such that the vessel is full again when leaving the port. The loaded cargo goes either to Port $2C + 1$ or the final Port $P = 2C + p + 1$. Below we will argue that this implies that at Ports 2 to $2C$, exactly one full lane will be unloaded and reloaded. The specific entries in the matrix R for Step 2 are presented here:

$$\begin{array}{ccccc} \begin{array}{r} r_{2,2C+1} = n \\ r_{3,P} = n+1 \\ r_{4,2C+1} = n+2 \end{array} & \cdots & \begin{array}{r} r_{C,2C+1} = n+C-2 \\ r_{C+1,P} = n+C-1 \\ r_{C+2,2C+1} = n+C-2 \end{array} & \cdots & \begin{array}{r} r_{2C-1,P} = n+1 \\ r_{2C,2C+1} = n \end{array} \end{array}$$

In Step 3, the vessel visits the Ports $2C + 1$ and onwards where a container going from Port i to Port j in the STACKING-NO-SHIFT instance is represented by a cargo item that is loaded in Port $2C + i$ and unloaded in Port $2C + j$ in the RoRo-SLOT-NO-SHIFT instance: $r_{2C+i,2C+j} = s_{i,j}$. Please note that the final destination for the cargo loaded in Step 3 is not the final Port P since $2C + j \leq 2C + p \leq P - 1$.

We now show that the STACKING-NO-SHIFT instance is a yes-instance if and only if the RoRo-SLOT-NO-SHIFT instance is a yes-instance. We start with the if-direction assuming that there is a plan for transporting the cargo in the RoRo-SLOT-NO-SHIFT instance with no shifts.

An area of the deck is said to be *connected to the stern* if there exists a path within the area from every slot in the area to the stern. The deck of the vessel is full after leaving each of the Ports 1, 2, 3, ..., $2C$, so for every Port 2 to $2C$ the cargo to be unloaded has to occupy an area of the deck connected to the stern. Every cargo item loaded in Step 1 will be unloaded at the ports in Step 2.

We claim that all the cargo with destination $C + 1$ must be placed in the longest lane of the deck (that has length $L = n + C - 1$). The top slot in this lane must be occupied by cargo going to Port $C + 1$. If this is not the case, then we have some destination port unloading less than L cargo, i.e., the cargo is not placed in an area connected to the stern producing a contradiction. If there is a cargo item with another destination than $C + 1$ in another slot of the longest lane, then we have another contradiction for cargo going to Port $C + 1$. With a similar argument we conclude that cargo to Port C is placed in one of the two lanes with length $L - 1$. As we go on, we can see that every lane is filled with cargo for exactly one destination when the vessel leaves Port 1. We have illustrated this argument in Fig. 2(b) that shows how the deck configuration could look after leaving Port 1 for $C = 4$, $p = 5$ and $n = 5$.

We now consider the details in Step 2 and look at the deck configuration after leaving Port $2C$. At this point in time, the odd-numbered lanes will contain cargo for Port $2C+1$ and the even-numbered lanes will contain cargo for the final Port P as illustrated in Fig. 2(c).

After unloading cargo in Port $2C+1$, the deck of the vessel is essentially a set of C LIFO columns since the even-numbered lanes are blocked with cargo for the final port. The loading/unloading plan for the containers in the Stacking-No-Shift instance at the Ports 1 to p is identical to the plan in the RoRo-Slot-No-Shift instance for loading and unloading cargo at the Ports $2C+1$ to $2C+p$. There are no shifts for the RoRo-Slot-No-Shift plan for the Ports $2C+1$ to $2C+p$ so this plan can easily be turned into a plan for handling the containers in the Stacking-No-Shift instance using C vertical LIFO columns with no shifts. The corresponding Stacking-No-Shift instance is in other words a yes-instance.

The only-if direction is easier to show. Assume that a plan exists for handling the containers in the Stacking-No-Shift instance with no shifts. We now use the RoRo-Slot-No-Shift plan as above illustrated by Fig. 2(b) and Fig. 2(c). From Port $2C+1$ and onwards, we follow the loading/unloading pattern from the Stacking-No-Shift plan. This is a plan with no shifts so the RoRo-Slot-No-Shift instance is a yes-instance. This concludes the proof for even values of C.

If C is odd, the proof is only slightly changed. The only difference is that we now have to use the longest lane of the deck as a column for simulating stacking of containers. This can easily be achieved by small changes to Step 2 of the proof. E.g., $r_{C+1,P} = n + C - 1$ shall be changed to $r_{C+1,2C+1} = n + C - 1$. □

It is worth noting that the only property used for the reduction in the proof of Theorem 1 is that no even-numbered lane has the same length as an odd-numbered lane. Consequently, the theorem holds for any deck configuration satisfying this property.

Unger [13] has presented a result stating that the interval overlap coloring problem mentioned earlier is $\mathcal{NP}$-complete for a fixed value $C = 4$. As far as we know, this result has not been questioned but a couple of Ungers related results have later been challenged [2,9]. If the result with $C = 4$ holds, then Theorem 1 holds for a fixed value $W = 7$. We have decided to state a weaker version of Theorem 1 with W as part of the input because that makes it possible for us to base our proof on the papers by Avriel et al. [1] and Garey et al. [4].

4 Generalization to arbitrary lane lengths

The proof in Section 3 required a specific layout of the deck to arrive at a configuration of the deck where exactly every second lane is filled with cargo for the final port and the remaining lanes contain no cargo. In this section we describe a slightly more complicated reduction from a Stacking-No-Shift instance to a RoRo-Slot-No-Shift instance with arbitrary lane lengths, with the only requirement that the minimum lane length is at least the total number of containers in the Stacking-No-Shift instance. The reduction also works, e.g., for rectangular decks.

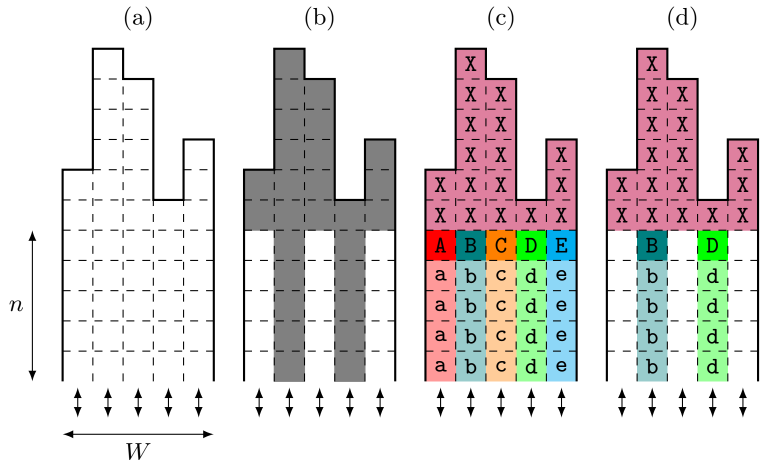


Fig. 3. Reduction with an arbitrary vessel deck: (a) Vessel deck for $n = 5$ and $W = 5$, (b) goal configuration for reduction, (c) state after $5W + 3$ ports, (d) obtained state for reduction.

Assume we want to solve a Stacking-No-Shift problem (S, C) with C columns, $S = \{s_{i,j}\}$ being a $p \times p$ 0-1-matrix, where p is the number of ports, and the total number of containers is $n = \sum s_{i,j}$. For the reduction in this section we only need to assume that the deck has $W = 2C - 1$ lanes, each of length at least n. See Fig. 3(a). We let $N \geq nW$ denote the total cargo capacity of the deck. Our RoRo-Slot-No-Shift instance will require $P = 5W + 5 + p$ ports.

Like in Section 3, the idea is to establish C lanes of length n separated by lanes only containing cargo heading for the final port, and use these lanes for the reduction, see Fig. 3(b). To achieve this, we will first establish through loading and unloading cargo at a sequence of ports a specific configuration with $2W + 1$ *types* of cargo, see Fig. 3(c): For each of the W lanes the $n - 1$ first slots are for a specific type cargo identified by W letters `a`, `b`, `c`, ..., the n'th slot is of type equal to the corresponding capital letters `A`, `B`, `C`, ..., and the remaining slots at the top of each lane are filled with cargo of type `X`. The letters will appear alphabetically left-to-right or right-to-left. Finally, by deleting all cargo of types `A`, `a`, `C`, `c`, `E`, `e`, ... we will have the desired configuration, see Fig. 3(d). Think of the type of a container as the color of the containers to load and unload. Whenever a type of container is loaded/unloaded, then all containers of this type is loaded/unloaded. At the end of the section we discuss how to convert the color view into the required $P \times P$ matrix $R = \{r_{i,j}\}$. It will be an invariant between ports, that there is zero or one cargo on the deck of each of the types `A`, `B`, `C`, ..., zero or $n - 1$ cargos of each of the types `a`, `b`, `c`, ..., and $N - nW$ cargo of type `X`, except for the initial and final empty deck.

To establish the configuration in Fig. 3(c), we consider two steps. The first step consists of $1 + 3W$ ports. At the first port we fill the deck with cargo with the required number of each type. At the following ports in the first step we unload all cargo of particular types and reload new cargo at the immediately following ports of the same type to enforce the initial loading of cargo at the first port to be in a specific configuration.

Port 1: Load $N - nW$ cargos of type `X`, one cargo of each of the W types `A`, `B`, `C`, ..., and $n - 1$ cargos of each of the W types `a`, `b`, `c`, ...
Port 2: Unload all cargo of type `a` and `A`.
Port 3: Load one cargo of type `A`.
Port 4: Load $n - 1$ cargos of type `a`.
Ports 5 to $3W + 1$ Repeat Ports 2 to 4 for each of the remaining $W - 1$ letters.

We now argue that to serve the above requests, the vessel must be loaded in the configuration shown in Fig. 3(d) after Port $3W + 1$, up to a permutation of the columns with letters. We denote the slots at the stern to be in *row* 1, the next slot in each lane to be in row 2, etc.

- After Port 1 the vessel is fully loaded, since we loaded $W(1+(n-1))+(N-nW) = N$ cargo.
- At Port 2 exactly n cargos are unloaded, which are connected to the stern. Since the vessel is full, we can only unload from the first n rows of the vessel. At Port 3 and Port 4 cargo of the sames types are loaded again, to make the vessel fully loaded. Similarly, for the other cargo types `b/B`, `c/C`, `d/D`, ...
- Since we at Ports 2 to $3W + 1$ unload nW different cargo that must be in the first n rows, it follows that we have unloaded exactly all cargo from the first n rows, i.e., all cargo of type `X` must reside in rows $n + 1$, $n + 2$, ...
- In row n there are cargo of type being a lower or capital letter. Assume it is cargo of type `a` or `A`. If a cargo is unloaded from this type, it is unloaded as part of exactly n cargo at Port 2 (or similar port for another letter), i.e., at Port 2 we unload exactly the first n cargo of one lane. Since we at Port 3 force cargo of type `A` to be loaded into the lane before all cargo of type `a` is loaded at the next port, `A` must be placed in row n after Port 4. It follows that there is exactly one cargo of type `A` in row n and none of type `a`.
- By repeating the above argument iteratively for the next letter in row n, it follows that there exists one lane with $n - 1$ cargo of type `a` followed by one cargo of type `A` in row n, and similarly for `b` and `B`, `c` and `C`, ... The remaining rows of the W lanes are filled with cargo of type `X`, i.e., the vessel is in the configuration shown in Fig. 3(c) after Port $3W + 1$, except that columns with letters may be permuted.

In the second step we add further load and unload requests, to force the columns to be alphabetically ordered from left-to-right or right-to-left after Port $3W + 1$.

Port $3W + 2$**:** Unload all cargo of type `a`, `A` and `B`.
Port $3W + 3$**:** Load one cargo of type `B`.
Port $3W + 4$**:** Load one cargo of type `A`.
Port $3W + 5$**:** Load $n - 1$ cargos of type `a`.

Port $3W + 6$**:** Unload all cargo of type `b`, `B`, `C`.
Ports $3W + 7$ **to** $4W + 3$**:** Unload the single cargos of type `D`, `E`, ... in alphabetical order, one cargo per port.
Ports $4W + 4$ **to** $5W + 2$**:** Load the $W - 1$ cargos of type ..., `D`, `C`, `B` in reverse alphabetical order, one cargo per port.
Port $5W + 3$**:** Load $n - 1$ cargos of type `b`.

We now argue that these load and requests force the columns to be alphabetically ordered, either from left-to-right or right-to-left.

– Since the cargo of type `A` and `B` are in row n, the unloading at Port $3W + 2$ forces `A` and `B` to be adjacent in row n. The subsequent loads at Ports $3W+3$ to $3W + 5$ reestablish a full vessel with cargo of the same type in all slots.
– Similarly, the request at Port $3W + 6$ forces the cargo of type `B` and `C` to be adjacent in row n, i.e., the type `B` cargo is between cargo of type `A` and `C` in row n.
– The unloading at Ports $3W + 7$ forces the cargo of type `C` and `D` to be adjacent, and the subsequent ports that type `D` and `E` cargo are adjacent, ... It follows that we have adjacent cargos of type (`A`, `B`), (`B`, `C`), (`C`, `D`), (`D`, `E`), ... in row n, which is only possible if the cargo in row n is ordered alphabetically from left-to-right or right-to-left.
– Since the unloaded cargo forms a path on the deck, the remaining Ports $4W+4$ to $5W + 2$ reload cargo in row n in reverse order of type of how it was unloaded, and Port $5W + 3$ reloads the remaining of the column with `B`. It follows that after Port $5W + 3$, we can only have the configuration in Fig. 3(c).

To arrive at the desired configuration required by the reduction from Section 3, we unload the first n cargo from every second lane, i.e., C lanes.

Port $5W + 4$**:** Unload all cargo of type `a`, `A`, `c`, `C`, `e`, `E`, ... (every second letter) for a total of nC cargo to unload.

We can for the following p ports now reduce an Stacking-No-Shift instance $\{s_{i,j}\}$ for C columns and n containers to an RoRo-Slot-No-Shift instance on the C empty lanes as described in Section 3 using p ports.

Ports $5W + 5$ **to** $5W + 4 + p$**:** For all $1 \leq i \leq p$ and for all $1 \leq j \leq p$, where $s_{i,j} = 1$, load a cargo at Port $5W + 4 + i$ heading for Port $5W + 4 + j$.

At the final port we unload all remaining cargo on the deck.

Port $5W + 5 + p$**:** Unload all cargo of type `X` and all cargo of type `b`, `B`, `d`, `D`, `f`, `F`, ... (every second letter) for a total of $N - nC$ cargo to unload.

In total the reduction from a Stacking-No-Shift problem requires $P = 5W + 5 + p$ ports, where $N + nW + 2n + W - 1$ cargo is loaded at Ports 1 to

$5W+4$ for setting up for the simulation, plus n cargo is loaded at ports $5W+5$ to $5W+4+p$ during the simulation itself.

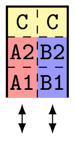
Since the RoRo-Slot-No-Shift problem is defined in terms of the matrix $R = \{r_{i,j}\}$, we briefly discuss how to construct cargo requests $r_{i,j}$ from Port i to Port j from the above description. Since cargo of type `X` is the only cargo from Port 1 to Port $P = 5W+5+p$, we have $r_{1,P} = N - nW$. For the remaining types we have $r_{1,2} = r_{1,5} = \cdots = r_{1,3W-1} = n$, since all initially loaded cargo of type `a` and `A` are unloaded at Port 2, and three ports later all cargo of types `b` and `B`, etc. The next requests for cargo of type `A`, `B`, ... is $r_{3,3W+2} = r_{6,3W+2} = 1$ and $r_{9,3W+6} = r_{12,3W+7} = \cdots = r_{3W,4W+3} = 1$, since `A` and `B` are unloaded at Port 3W+2, whereas the remaining types `C`, `D`, ... are unloaded at Ports $3W+6$, $3W+7$, ..., $4W+3$, respectively. The cargo of `a`, `b`, `c`, ... loaded at Ports 4 to $3W+1$ is unloaded at Ports $3W+2$, $3W+6$, $5W+4$, P, $5W+4$, $P, \ldots, 5W+4$, respectively, i.e., $r_{4,3W+2} = r_{7,3W+6} = r_{10,5W+4} = r_{13,P} = r_{16,5W+4} = r_{19,P} = \cdots = r_{3W+1,5W+4} = n-1$. This covers all cargo loaded at the first $3W+1$ ports. The cargo loaded at Ports $3W+2$ to $5W+4$ can be handled similarly. For the reduction itself we have $r_{5W+4+i,5W+4+j} = s_{i,j}$, for $1 \leq i < j \leq p$. It follows similar to the proof of Theorem 1 that that the RoRo-Slot-No-Shift instance is $\mathcal{NP}$-hard for all vessel decks of width $W \geq 7$, in particular for rectangular decks where all lanes have equal length.

5 Generation of RoRo instances requiring shifts

The existence of RoRo-Slot-No-Shift instances that cannot be handled without shifts is a corollary of Theorem 1. Actually, the reductions described in Section 3 and Section 4 can be used in a constructive way as recipes for generating such hard RoRo-Slot-No-Shift instances for the deck configurations considered in those sections. The recipe works as follows: start with a Stacking-No-Shift instance requiring shifts and generate the r-matrix specified in the reduction. This will produce a RoRo-Slot-No-Shift instance requiring shifts. The following is just one example for a hard Stacking-No-Shift instance to use as the starting point: $p = 8$, $C = 3$, $s_{1,5} = s_{2,6} = s_{3,7} = s_{4,8} = 1$ (all other s-entries can as an example be set to 0). This particular Stacking-No-Shift instance is hard since there are more containers than stacks and none of the containers can be put in the same stack without requiring shifts.

There are other ways to produce hard RoRo-Slot-No-Shift instances for small deck configurations. Fig. 4 illustrates a RoRo-Slot-No-Shift instance, that cannot be served without a shift of a cargo. Cargo requests at Ports 1–7 establish the configuration in Fig. 4(a), up to a permutation of the `A` and `B` columns, provided no shift of already loaded cargo is allowed (the argument for the `A` and `B` columns is similar as in Section 4). The unload request at Port 8 cannot unload the type `C` cargo without shifts, since `A2` and `B2` are blocking after unloading `A1` and `B1`. To serve the request it is sufficient at Port 8 after `A1` and `B1` are unloaded to perform a shift of `B2` to below `A2` on the deck.

(a)



(b)

Port 1: Load `C`, `C`, `A2`, `B2`, `A1`, `B1`
Port 2: Unload `A1`, `A2`
Port 3: Load `A2`
Port 4: Load `A1`
Port 5: Unload `B1`, `B2`
Port 6: Load `B2`
Port 7: Load `B1`
Port 8: Unload `A1`, `B1`, `C`, `C`
Port 9: Unload `A2`, `B2`

(c)

$r_{1,2} = r_{1,5} = r_{1,9} = 2$
$r_{3,9} = r_{4,8} = r_{6,9} = r_{7,8} = 1$

Fig. 4. A RoRo-Slot-No-Shift instance that cannot be served without shifting cargo in a port. (a) deck layout and configuration obtained by requests at Ports 1–7 (if no shifting is performed). (b) Cargo types loaded and unloaded at the different ports. (c) The resulting non-zero entries of R.

6 A greedy algorithm for special RoRo instances

In this section we complement the previous hardness results with a greedy algorithm for the RoRo-Slot-No-Shift problem for cases where the deck layout is a rectangle $W \times L$ with W ramps ($W \geq 1$ and $L \geq 1$), and at most $W \cdot L$ cargo is loaded at an arbitrary number of source ports (possibly more than W) before it is unloaded at subsequent $P \leq W$ destination ports. For this case we show that there exists a fixed assignment of destination ports to the slots, where the number of slots assigned for a destination port equals the amount of cargo for the destination port. The cargo can be unloaded in any arbitrary order of the P ports. In particular, we show that all cargo for a port p is connected to a ramp via slots all with cargo for port p. See Fig. 5(b). When loading cargo for a destination port, the cargo is placed as far away as possible from the loading ramp in the area assigned for this destination port (largest possible breadth-first-search (BFS) distance). Unloading at a destination port is performed greedily in reverse order in the area assigned to the destination port.

To create the assignment, we first identify all destination ports with exactly L cargo (`A` and `B` in Fig. 5(b)). These are assigned one lane each, left-to-right. We denote the remaining destination ports with $\geq L + 1$ and $\leq L - 1$ destination cargo as *heavy* and *light* ports, respectively. We next consider the heavy ports in arbitrary order. Consider the first heavy port p. Assign the full leftmost lane for port p (e.g., `C`), and try to assign the bottom of the next lane to the next light port q (e.g., `D`), and fill the top-part with further slots for p. If all slots in the lane get assigned, continue repeatedly filling port p and the next light port into the next lane. Otherwise, the remaining slots for p (possibly none) and q cannot occupy the full lane (like `C` and `E` cannot fill the 5th leftmost lane). Cancel inserting q in the lane, and fill the lane with next heavy port p' (e.g., `G`), and insert p' top-down in the next column with q at the bottom. At least $L + 1$ connected slots are assigned to the heavy port p' in the two columns. If p' and q again cannot fill the lane, we cancel insertion q and proceed repeatedly with the next heavy port. While there are still heavy and light ports left, this ensures

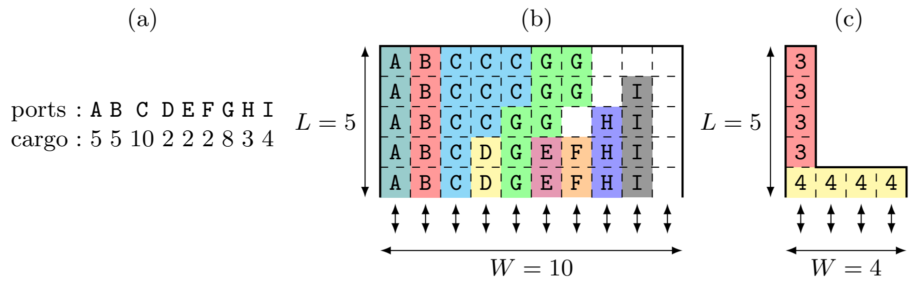


Fig. 5. (a–b) Greedy algorithm for $P = 9$ destination ports. (c) Cargo requiring shifts.

columns left-to-right are completely filled by cargo for at most two destination ports, each lane starts with a new destination port at the bottom, and all slots assigned to a given destination port are connected. Two final cases remain: If there are no more heavy ports, the remaining light ports are each assigned the bottom part of one lane (possible since $P \le W$; see F, H and I in Fig. 5(b)). If there are no light ports, the heavy ports are just assigned to the remaining lanes left-to-right and top-down.

Without the rectangular deck assumption, solutions are not guaranteed to exist. Assume a deck has $W \ge 2$ lanes of lengths $W + 1, 1, \ldots, 1$, and we have cargo requests $r_{1,3} = r_{2,4} = W$ from two sources ports to two destination ports, see Fig. 5(c). It is easy to verify that this problem has no solution with no shifts, since one cargo for Port 3 must be placed at the top of the longest lane to be able to load all cargo, but this cargo cannot be unloaded without shifts.

7 Conclusion

In this paper, we investigated the complexity of stowage planning and delved into the specific case of slot-based RoRo stowage planning. We reduced a stacking problem to a slot-based RoRo stowage problem by dividing the deck space into a set of LIFO lanes. With the transformation, we showed that the simplest form of the RoRo stowage problem with identical cargo is $\mathcal{NP}$-complete using a reduction from the Stacking-No-Shift problem. Further, we presented a version of the problem with arbitrary lane length that is $\mathcal{NP}$-hard, and showed how these complexity results can be used to generate hard instances for the problem. Finally, for the RoRo problem where the deck is a rectangle with at least at many lanes as destination ports for cargo and where all cargo is loaded at ports before unloading at subsequent ports, we give a greedy algorithm that always finds a plan with no shifts if the total amount of cargo fits the deck.

References

1. Avriel, M., Penn, M., Shpirer, N.: Container ship stowage problem: complexity and connection to the coloring of circle graphs. Discrete Applied Mathematics **103**(1–3), 271–279 (2000). https://doi.org/10.1016/S0166-218X(99)00245-0

2. Bachmann, P.: 3-coloring Circle Graphs in Theory and Practice. Master's thesis, University of Passau, Germany (2022), `https://www.fim.uni-passau.de/fileadmin/dokumente/fakultaeten/fim/lehrstuhl/rutter/abschlussarbeiten/2022-Patricia_Bachmann_MA.pdf`
3. Cornelsen, S., Stefano, G.D.: Track assignment. Journal of Discrete Algorithms **5**(2), 250–261 (2007). `https://doi.org/10.1016/j.jda.2006.05.001`
4. Garey, M.R., Johnson, D.S., Miller, G.L., Papadimitriou, C.H.: The complexity of coloring circular arcs and chords. SIAM Journal on Algebraic Discrete Methods **1**(2), 216–227 (June 1980). `https://doi.org/10.1137/0601025`
5. 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: Paias, A., Ruthmair, M., Voß, S. (eds.) Computational Logistics. pp. 35–49. Springer International Publishing, Cham (2016). `https://doi.org/10.1007/978-3-319-44896-1_3`
6. Hvattum, L.M., Fagerholt, K., Armentano, V.A.: Tank allocation problems in maritime bulk shipping. Computers & Operations Research **36**(11), 3051–3060 (November 2009). `https://doi.org/10.1016/j.cor.2009.02.002`
7. Jansen, K.: The mutual exclusion scheduling problem for permutation and comparability graphs. Information and Computation **180**(2), 71–81 (2003). `https://doi.org/10.1016/S0890-5401(02)00028-7`
8. Jia, B., Fagerholt, K., Reinhardt, L.B., Rytter, N.G.M.: Stowage planning with optimal ballast water. In: Lalla-Ruiz, E., Mes, M., Voß, S. (eds.) Computational Logistics, Lecture Notes in Computer Science, vol. 12433, pp. 84–100. Springer Nature Switzerland, Cham (2020). `https://doi.org/10.1007/978-3-030-59747-4_6`
9. König, F.G., Lübbecke, M.E.: Sorting with complete networks of stacks. In: Hong, S., Nagamochi, H., Fukunaga, T. (eds.) Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5369, pp. 895–906. Springer, Berlin, Heidelberg (2008). `https://doi.org/10.1007/978-3-540-92182-0_78`
10. Puisa, R.: Optimal stowage on Ro-Ro decks for efficiency and safety. Journal of Marine Engineering & Technology **20**(1), 17–33 (January 2021). `https://doi.org/10.1080/20464177.2018.1516942`
11. Tang, L., Liu, J., Yang, F., Li, F., Li, K.: Modeling and solution for the ship stowage planning problem of coils in the steel industry. Naval Research Logistics (NRL) **62**(7), 564–581 (2015). `https://doi.org/10.1002/nav.21664`
12. Tierney, K., Pacino, D., Jensen, R.M.: On the complexity of container stowage planning problems. Discrete Applied Mathematics **169**(0), 225–230 (2014). `https://doi.org/10.1016/j.dam.2014.01.005`
13. Unger, W.: On the k-colouring of circle-graphs. In: Cori, R., Wirsing, M. (eds.) STACS 88, 5th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 11-13, 1988, Proceedings. Lecture Notes in Computer Science, vol. 294, pp. 61–72. Springer (1988). `https://doi.org/10.1007/BFb0035832`
14. Van Twiller, J., Sivertsen, A., Pacino, D., Jensen, R.M.: Literature survey on the container stowage planning problem. European Journal of Operational Research pp. 841–857 (December 2023). `https://doi.org/10.1016/j.ejor.2023.12.018`
15. Vilhelmsen, C., Larsen, J., Lusby, R.: A heuristic and hybrid method for the tank allocation problem in maritime bulk shipping. 4OR **14**(4), 417–444 (2016). `https://doi.org/10.1007/s10288-016-0319-x`
16. Øvstebø, B.O., Hvattum, L.M., Fagerholt, K.: Optimization of stowage plans for roro ships. Computers & Operations Research **38**(10), 1425–1434 (2011). `https://doi.org/10.1016/j.cor.2011.01.004`