European Journal of Operational Research 235 (2014) 387–398

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Exact and heuristic methods for placing ships in locks

J. Verstichel [a,c,*], P. De Causmaecker [b,c], F.C.R. Spieksma [d], G. Vanden Berghe [a]

[a] CODeS, KAHO Sint-Lieven, Gebroeders De Smetstraat 1, 9000 Gent, Belgium
[b] CODeS, Department of Computer Science, KU Leuven, ITEC-iMinds, Belgium
[c] KU Leuven campus Kortrijk, Etienne Sabbelaan 53, 8500 Kortrijk, Belgium
[d] ORSTAT, KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium

ARTICLE INFO

Article history:
Received 2 November 2012
Accepted 25 June 2013
Available online 3 July 2013

Keywords:
Ship placement problem
Packing
Heuristics
Lock scheduling
Decomposition

ABSTRACT

The ship placement problem constitutes a daily challenge for planners in tide river harbours. In essence, it entails positioning a set of ships into as few lock chambers as possible while satisfying a number of general and specific placement constraints. These constraints make the ship placement problem different from traditional 2D bin packing. A mathematical formulation for the problem is presented. In addition, a decomposition model is developed which allows for computing optimal solutions in a reasonable time. A multi-order best fit heuristic for the ship placement problem is introduced, and its performance is compared with that of the left-right-left-back heuristic. Experiments on simulated and real-life instances show that the multi-order best fit heuristic beats the other heuristics by a landslide, while maintaining comparable calculation times. Finally, the new heuristic's optimality gap is small, while it clearly outperforms the exact approach with respect to calculation time.

1. Introduction

When entering or leaving a port, ships often pass one or more locks. So do barges travelling on a network of waterways. The locks provide a constant water level for ships while loading or unloading at the docks, or they control the flow and the level of inland waterways.

The growing number of container shipments causes high demands on sea ports (Wiese, Suhl, & Kliewer, 2010). Improving the ship handling can reduce their *time in port* and make a seaport economically more attractive, engendering a strong competition between (geographically close) seaports (Bish, 2003; Chen, Bostel, Dejax, Cai, & Xi, 2006; Cullinane & Khanna, 2000; Günther & Kim, 2006). While many aspects of handling ships and containers in seaports have been extensively researched (Stahlbock & Voß, 2008), one key component of the port's infrastructure has received little attention: the locks. A suboptimal usage of the locks' capacity can however strongly increase the handling times of ships. When the lock is unable to transfer a given ship in time, this ship could miss its time window at the terminal, leading to a strong increase in total time in port and a reduced efficiency of the terminal. Improved operation of these locks can therefore play an important role in increasing a port's efficiency and economical attractiveness.

The expected increase of intermodal transport is a major incentive for improving lock efficiency on inland waterways (European Commission, 2009, 2011). Intermodal transport is the combination of multiple transport modes in a single transport chain without a change of container for the goods. Inland navigation is a most promising transport mode in the intermodal chain, with its availability of access capacity in the network and environmentally friendly character as most important benefits. This is especially true in the Belgian and Western-European context, where inland waterways play a crucial role in the hinterland access of major sea ports (Notteboom & Rodrigue, 2005). Increasing the efficiency of (intermodal) barge transport through better lock operations is therefore a key issue in supporting future freight flows and increasing the market share of inland navigation.

2. Literature review

Only a small number of academic papers focus on lock planning. Wilson (1978) investigates the applicability of different queuing models for lock capacity analysis. The research shows that good queuing models exist for single chamber locks, but not for locks with parallel chambers. Some of the other research has focussed on the Upper Mississippi River (UMR). On that river, barges are joined together into tows for transport, which then need to be transferred by single chamber locks that are often smaller than the tow itself. The tow is split into different groups of barges and these groups are transferred one at a time, after which they are rejoined for the next phase of their travel. Nauss (2008) presents

* Corresponding author at: CODeS, KAHO Sint-Lieven, Gebroeders De Smetstraat 1, 9000 Gent, Belgium. Tel.: +32 9 265 87 04.
E-mail address: jannes.verstichel@kahosl.be (J. Verstichel).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.06.045

Nomenclature

Sets

N	set of ships that need to be placed, $N = \{1,2,\ldots,n\}$
M	set of lockages (bins) available, $M = \{1,2,\ldots,m\}$, where m should be a sufficiently large number, e.g. $m = n$ or equal to an appropriate upper bound
$MOOR_i$	set of ships to which ship $i \in N$ is allowed to moor

Parameters

W, L	width and length of the chamber (integer)
w_i, l_i	width and length of ship $i \in N$ (integer)
dF_i, dB_i	minimal distance between ship $i \in N$ and the front/back of the chamber
sW_{ij}, sL_{ij}	minimal safety distance between ships $i \in N$ and $j \in N$ when they are adjacent, or laying behind each other
$ship_0$	represents the left quay of the chamber: $x_0 = 0$, $y_0 = 0$, $w_0 = 0$, $l_0 = L$
$ship_{n+1}$	represents the right quay of the chamber: $x_{n+1} = W$, $y_{n+1} = 0$, $w_{n+1} = 0$, $l_{n+1} = L$
K	relative cost of the number of lockages

Variables

x_i, y_i	integer variables that define the x and y position of ship $i \in N$ in the chamber (lower left corner of ship)
$left_{ij}$	binary variable that indicates if ship $i \in N$ is left to ship $j \in N$ ($left_{ij} = 1 \Rightarrow x_i + w_i \leqslant x_j$)
b_{ij}	binary variable that indicates if ship $i \in N$ is behind ship $j \in N$ ($b_{ij} = 1 \Rightarrow y_i + l_i \leqslant y_j$)
ml_{ij},mr_{ij}	binary variables that indicate if ship $i \in N$ is moored to ship $j \in N$'s left, respectively right, side
z_k	binary variable that indicates whether lockage $k \in M$ is used or not
f_{ik}	binary variable that indicates whether ship $i \in N$ is processed in lockage $k \in M$ or not
v_{ij}	binary variable, 0 when ship $i \in N$ and $j \in N$ are processed in the same lockage, 1 otherwise

optimal sequencing of tows/barges for single chamber locks with set-up times. The approach allows one tow/barge to be transferred at a time. Furthermore, it considers all tows/barges to be present at the lock before the first lockage. A simulation model for comparing different strategies to relieve congestion problems on the Upper Mississippi river is presented in Smith, Sweeney, and Campbell (2009). The strategies aim at increasing the throughput of the locks and a simulation tool was built for validating them. Smith et al. (2011) further increases the performance of these locks using more complex decision rules based on heuristics and MIP models.

Another part of the lock scheduling literature focusses on locks with (parallel) chambers capable of transferring several ships together. A recent contribution (Verstichel & Vanden Berghe, 2009) deals with planning a lock with three parallel chambers, two of which are identical. The presented approach allows more than one ship to be transferred in one chamber at the same time, and considers independent operation of the chambers. A problem specific left-right-left-back heuristic combined with a late acceptance hill-climber generated good quality results. Scheduling the chamber operations of a lock with identical parallel chambers is investigated in Verstichel, De Causmaecker, and Vanden Berghe (2011). The contribution identifies the problem as the identical parallel machine scheduling problem with sequence dependent setup times and release dates, and presents a mathematical model. Both inland locks with fixed processing times and sea port locks with ship dependent processing times are considered. A meta-heuristic approach to the problem is presented, and its performance is compared with that of the first-come-first-served decision rule. Coene, Spieksma, and Vanden Berghe (2011) focus on planning lock operations without considering the actual placement of ships within the chamber.

The present paper considers locks with at least one, but possibly multiple parallel chambers with different properties. Each chamber has a limited capacity, based on its dimensions, and a certain lockage duration, i.e. the time needed to change the water level in the chamber from the level on one side to the level on the other side. Chambers with identical dimensions and lockage durations are of the same chamber type. Therefore, when some of the chambers are identical, a lock will consist of fewer chamber types than the number of chambers. When planning transport through a lock, different issues may arise. The time needed to transfer a number of sea ships, for example, depends on their size and manoeuvrability. Positioning the ships in the chamber may even take longer than the actual lockage operation of the chamber, particularly when transferring large ships that require tugboats. When only inland ships are transferred, the time needed to position the ships can be considered constant as these ships are much smaller and can easily be positioned inside the chamber. The time needed to position the ships inside the chamber could even be included in the lockage time of the chambers. Also strategies for operating parallel chambers differ. Some locks use paired chambers, which must be operated simultaneously, while other systems allow independent operation of the chambers. In spite of these different approaches, all aforementioned cases require solutions to the ship placement problem. When several ships can be transferred simultaneously in one lockage, a better ship placement procedure may strongly increase the lock's efficiency. Two examples where locks play a crucial role in the transportation chain are the Port of Antwerp and the Albertkanaal in Belgium.

This paper introduces a mathematical model for the ship placement problem and compares three algorithms: (i) an exact decomposition based integer programming approach, (ii) a modified left-right-left-back heuristic (Verstichel & Vanden Berghe, 2009), and (iii) a multi-order best fit heuristic. The main contribution of part (i) is the in-depth analysis of the ship placement model and its characteristics, and the introduction of a decomposition method generating optimal solutions, even for very large instances.

Section 3 introduces the details of the problem and a mathematical model for the ship placement problem. Section 4 describes the three solution approaches that were applied to the problem. The experimental setup and results are discussed in Section 5, followed by the conclusions in Section 6.

3. Problem definition and model

3.1. Problem definition

Ships constitute the first major component of the ship placement problem. They are characterised by a width w_i, and a length ℓ_i. By assuming rectangular-shaped ships, we simplify the evaluation of the placement constraints. This simplification is common practice, as the exact shape of the ships is often not available to

lock operators. It is necessary to maintain a certain safety distance between ships when they are placed together in one chamber. These distances are defined for each pair of ships, based on the ships' dimensions, types and the number of tugboats used. The second component of the problem is the lock, which consists of a single chamber defined by its width W and length L. Certain distances must be satisfied between the ships and the chamber's doors, both for safety (collisions) and practical (mooring posts) reasons.

Given an ordered list of n ships, the ship placement problem aims at minimising the number of lockages needed to place all ships, subject to a number of specific placement and sequence constraints. The problem is reminiscent of the well known 2D bin packing problem (or 2D rectangular SBSBPP (Wäscher, Haußner, & Schumann, 2007)) where a set of rectangular items (ships) needs to be positioned inside as few rectangular bins (lockages) as possible, and where rotation of the items is not allowed. However, there are a number of relevant differences. First, in the ship placement problem, there is a sequence constraint stipulating that the ships need to be processed in a first-come-first-served (FCFS) way with respect to their position in the ship list. If we assume that the lockages are ordered by their index, and that the ship at position i in the ship list is positioned in lockage k, then the ships at position $j > i$ in this list are not allowed in any lockage with index $l < k$. Secondly, all ships should be placed in the first lockage they fit in. This means that if the optimal number of lockages is 5, and ship 7 can be placed in either lockage 3 or lockage 4, it should be placed in lockage 3. Although it may seem that these constraints have no immediate use for the ship placement itself, they are vital when connecting the ship placement problem to the timing part of the lock scheduling problem. The lockages generated by a ship placement algorithm can be used by scheduling algorithms for the generalised lock scheduling problem. If the obtained lockages are suboptimal from a scheduling point-of-view, the scheduling algorithm can change the ship order, effectively generating a new instance for the ship placement algorithm. By iterating between both solution methods, a globally better solution to the lock scheduling problem can be found. Without the sequence constraints, this kind of subproblem interaction would be much harder to accomplish.

The third difference is the set of mooring constraints: each ship must be moored either to the quay, or to another ship. Geometrically, ship i is said to be moored to ship j, when ship i is adjacent to ship j over its entire length. This constraint implies that each ship will be connected to the quay through larger ships, or through ships of equal size. It should be noted that the width of a ship does not affect the evaluation of the mooring constraint. Examples of solutions that do not satisfy this mooring constraint are visualised in Fig. 1(d)–(f), while the standard 2D bin packing constraints are visualised in Fig. 1(a)–(c).

While the above described mooring constraints are sufficient in most cases where only inland ships are considered, some additional mooring/safety constraints may be required in other settings. Sea ships, for example, can only be moored to the quay. In the case of mixed sea/inland traffic, inland ships are normally not allowed to moor to sea ships. Furthermore, in some locks ships cannot moor to each other when the difference in hull height above the water is too large (for example fully loaded and empty ships). These additional constraints can be modelled as group-mooring constraints that only allow mooring to ships that belong to a certain group.

The last set of additional constraints deals with the typical safety distances of the ship placement problem. Some 'room' must be given to ships, allowing for corrections to prevent collisions while manoeuvring in and out of the chamber, and in case of a possible minor accident during the lockage operations. While for an inland setting these safety distances can be considered equal for all ship tuples, traffic in sea ports requires a more individual

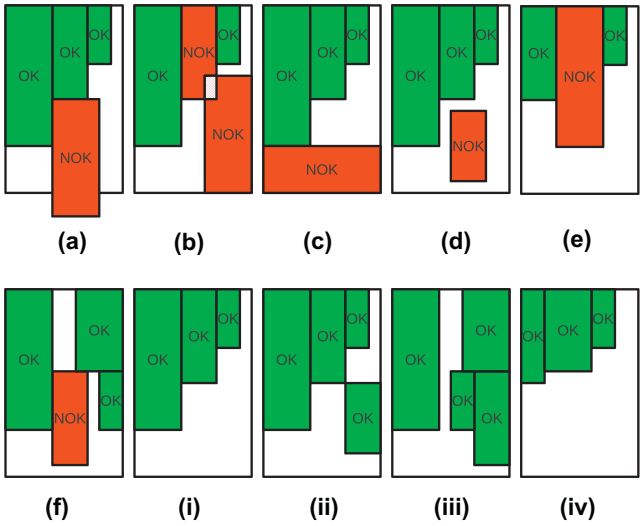

Fig. 1. A visual representation of the mooring constraints of the lock scheduling problem. (OK: ship placed in a correct way, NOK: ship violates a mooring constraint).

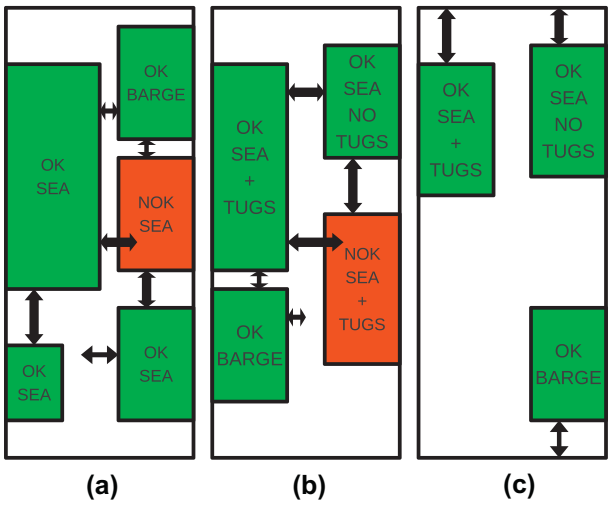

Fig. 2. A visual representation of the safety distance constraints of the lock scheduling problem. (OK: ship placed in a correct way, NOK: ship violates a safety distance constraint).

approach. The safety distance constraints for different ship tuples are visualised in Fig. 2. Based on the dimensions and type of both ships, a minimal lateral and longitudinal safety distance can be calculated (Fig. 2(a)). When both ships use tugboats, the lateral safety distance must also be sufficiently large to allow for the tugboats to sail between the ships when leaving the chamber before the lockage operations starts (Fig. 2(b)). A last distance is defined between each ship and the doors of the chamber (Fig. 2(c)). These are implied both for safety (i.e. avoid collisions with the doors) and practical reasons (position of the mooring poles to which the ship can moor). This distance depends on both the chamber type and the dimensions of the ship.

In the absence of the mooring and safety constraints, the ship placement problem is NP-hard. This follows from a result in Leung, Tam, Wong, Young, and Chin (1990) stating that it is NP-complete to decide whether a given set of squares fits in a given square.

One might wonder whether the mooring constraint really complicates matters, or in other words, whether instances exist for

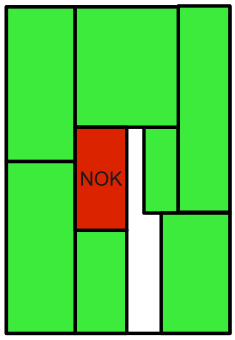

Fig. 3. No feasible single bin ship placement solution can be found for this problem, while constructing a single bin 2D bin packing solution is trivial. (NOK: ship in violation with the ship placement constraints).

which the traditional 2D bin packing problem requires fewer bins (lockages) than the ship placement problem. The answer to this question is positive, and we provide two instances with this property. The first instance is shown in Fig. 3. Here, the application of the ship placement constraints results in a solution with two lockages, while the 2D bin packing problem requires only one bin. Due to the limited number of ships, this example can be solved to optimality with the model from Section 4.2 in less than 100 s. The obtained solution confirms that two lockages are required to place all ships when the ship placement constraints are taken into account. The 8 items have the following dimensions: 12×3, 10×4, 9×4, 7×6, 7×4, 6×3, 6×3 and 5×2, while the chamber has dimension 19×13. The second instance considers a perfect packing where the items have to be placed in three rows to obtain an optimal single bin solution. All 14 rectangles have a width of 1, and their lengths are 27 (2 items), 12 (5 items), 10 (6 items) and 6 (1 item), while the chamber has a dimension of 60×3. There exist six different optimal solutions to the 2D bin packing problem for the given instance, each of which violates at least one ship placement constraint. It follows that when this instance is viewed as a ship placement problem, at least two lockages are required. Fig. 4 visualises these single bin solutions and highlights the items that are in violation with the ship placement constraints. For simplicity's sake, the group mooring and safety constraints are omitted in these examples. When taken into consideration, the difference between the optimal solutions would become even larger.

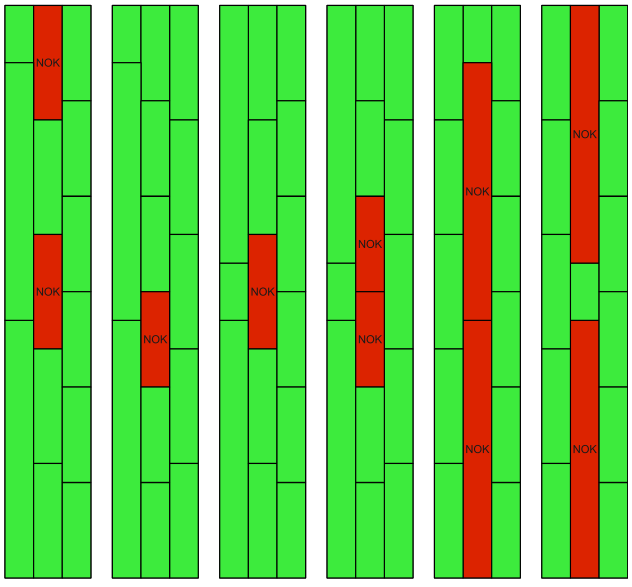

Fig. 4. The six optimal 2D bin packing solutions for this problem all violate at least one ship placement constraint. (NOK: ship in violation with the ship placement constraints).

3.2. MILP model

We present a Mixed Integer Linear Programming model for the ship placement problem, which is partially based on the model of Pisinger and Sigurd (2005). This model contains two additional ships ($i = 0$ and $i = n + 1$) that represent the left and right quay respectively. These ships allow for a straightforward implementation of the mooring constraints, as the quays can now be seen as large ships with a fixed position, to which a ship can be moored. The model also uses a set $MOOR_i$ that contains all ships to which ship i, $i = 1, \ldots, n$ is allowed to moor. In the inland setting, this set contains all ships that are longer than ship i, as a ship can only moor to ships that are at least equally long. In a more general setting, this set may contain only ships of a certain type. Sea ships, for example, may often moor only to the quay, while no ships are allowed to moor to a sea ship. By changing the ships that are available in the $MOOR_i$ sets, any such mooring restriction can be implemented in this model. The model includes safety distances between ships, and between each ship and the chamber doors. These distances depend on ship properties like the ship dimensions, ship types and whether or not the ship requires tugboats. While the lateral safety distance between a sea ship and a barge may average at 1.5 m, the minimal lateral distance between two sea ships, both requiring tugboats may be as large as 12 m.

The main objective (1) is to minimise the number of lockages required for placing all the ships ($\sum_{k \in M} z_k$). Among different solutions with an equal number of lockages, the one where all ships are placed in the lockage with the lowest possible index is favoured ($\sum_{k \in M} k \sum_{i \in N} f_{ik}$). In other words, when $ship_i$ can be placed in either $lockage_k$ or $lockage_{k+1}$, while the resulting solutions have the same total number of lockages, it should be placed in $lockage_k$. To ensure that the total number of lockages is always more important than the lockage in which a ship is placed, the main objective is multiplied by K, which should be sufficiently large, for example $K = |M|^2$.

$$\text{minimise} \quad K \sum_{k \in M} z_k + \sum_{k \in M} k \sum_{i \in N} f_{ik} \tag{1}$$

Constraints (2)–(4) ensure that two ships transferred in the same lockage do not overlap.

$$left_{ij} + left_{ji} + b_{ij} + b_{ji} + (1 - f_{ik}) + (1 - f_{jk}) \geqslant 1, \tag{2}$$

$$\forall i < j, \ i,j \in N, k \in M$$

$$x_i + w_i \leqslant x_j + W(1 - left_{ij}), \quad \forall i \neq j, \ i,j \in N \tag{3}$$

$$y_i + l_i \leqslant y_j + L(1 - b_{ij}), \quad \forall i \neq j, \ i,j \in N \tag{4}$$

All ships must be placed inside the dimensions of the chamber. This is modelled by constraints (5) and (6).

$$x_i + w_i \leqslant W, \quad \forall i \in N \tag{5}$$

$$y_i + l_i \leqslant L, \quad \forall i \in N \tag{6}$$

Constraint (7) models the safety distance between two adjacent ships, see Fig. 2. This safety distance depends on the properties of both ships.

$$x_j + (W + sW_{ij})(1 - left_{ij} + b_{ij}) \geqslant x_i + w_i + sW_{ij}, \quad \forall i \neq j, \ i,j \in N \tag{7}$$

The safety distance requirements for two ships that are laying behind each other are modelled in constraint (8) and visualised in Fig. 2.

$$y_j + (L + sL_{ij})(1 - b_{ij} + left_{ij}) \geqslant y_i + l_i + sL_{ij}, \quad \forall i \neq j,\ i,j \in N \tag{8}$$

Constraints (9) and (10) ensure that a minimal distance between a ship and the front and back door of the chamber is respected. This distance is determined by both the safety requirements and the lay-out of the mooring poles on the quays.

$$y_i \geqslant dF_i, \quad \forall i \in N \tag{9}$$

$$y_i + l_i \leqslant L - dB_i, \quad \forall i \in N \tag{10}$$

Constraint (11) ensures that each ship is transferred by exactly one lockage.

$$\sum_{k \in M} f_{ik} = 1, \quad \forall i \in N \tag{11}$$

Constraint (12) models that each lockage that transfers a ship must be used.

$$f_{ik} \leqslant z_k, \quad \forall i \in N, k \in M \tag{12}$$

Ship i can only moor to ship j's left side when it is contained within ship j's length. This is modelled by constraints (13) and (14).

$$y_j \leqslant y_i + (1 - ml_{ij})L, \quad \forall i \in N, j \in MOOR_i \tag{13}$$

$$y_i + l_i \leqslant y_j + l_j + (1 - ml_{ij})L, \quad \forall i \in N, j \in MOOR_i \tag{14}$$

Constraints (15) and (16) model the additional requirement for mooring ship i to ship j: they must be adjacent.

$$x_j \leqslant x_i + w_i + (1 - ml_{ij})W, \quad \forall i \in N, j \in MOOR_i \tag{15}$$

$$x_j \geqslant x_i + w_i - (1 - ml_{ij})W, \quad \forall i \in N, j \in MOOR_i \tag{16}$$

Ship i cannot only moor to the left side of another ship, it can also moor to the left side of the right quay (ship $n+1$). Here it suffices to check that ships i and $n+1$ are adjacent (Constraints (17) and (18)), because ship i will always be contained within the quay's length.

$$x_{n+t} \leqslant x_i + w_i + (1 - ml_{i,n+1})W, \quad \forall i \in N \tag{17}$$

$$x_{n+t} \geqslant x_i + w_i - (1 - ml_{i,n+1})W, \quad \forall i \in N \tag{18}$$

Ship i can only moor to ship j's right side when it is contained within ship j's length. This is modelled by constraints (19) and (20).

$$y_j \leqslant y_i(1 - mr_{ij})L, \quad \forall i \in N, j \in MOOR_i \tag{19}$$

$$y_i + l_i \geqslant y_j + l_j + (1 - mr_{ij})L, \quad \forall i \in N, j \in MOOR_i \tag{20}$$

Constraints (21) and (22) model the additional requirement for mooring ship i to ship j: they must be adjacent.

$$x_j + w_j \leqslant x_i + (1 - mr_{ij})W, \quad \forall i \in N, j \in MOOR_i \tag{21}$$

$$x_j + w_j \geqslant x_i - (1 - mr_{ij})W, \quad \forall i \in N, j \in MOOR_i \tag{22}$$

Ship i cannot only moor to the right side of another ship, it can also moor to the right side of the left quay (ship 0). Here it suffices to check that ships i and 0 are adjacent (Constraints (23) and (24)), because ship i will always be contained within the quay's length.

$$x_0 \leqslant x_i + (1 - mr_{i,0})W, \quad \forall i \in N \tag{23}$$

$$x_0 \geqslant x_i - (1 - mr_{i,0})W, \quad \forall i \in N \tag{24}$$

All ships have to be moored and this is modelled by constraint (25). A ship can be moored to another ship, the left quay or the right quay.

$$\sum_{j \neq i,\ j \in N} (ml_{ij} + mr_{ij}) + mr_{i,0} + ml_{i,n+1} \geqslant 1, \quad \forall i \in N \tag{25}$$

When two ships have the same length, they might end up moored to one another. This is not desirable, as this could leave both ships unattached to the quay. Constraint (26) disables this kind of 'fake' mooring.

$$ml_{ij} + mr_{ji} \leqslant 1, \quad \forall i \neq j,\ i,j \in N \tag{26}$$

Two ships that are transferred in different lockages cannot moor to each other. Constraints (27)–(29) ensure that the mooring constraints are only valid for two ships that are transferred in the same lockage.

$$f_{ik} - f_{jk} \leqslant v_{ij}, \quad \forall i < j, i,j \in N, k \in M \tag{27}$$

$$f_{jk} - f_{ik} \leqslant v_{ij}, \quad \forall i < j,\ i,j \in N, k \in M \tag{28}$$

$$ml_{ij} + mr_{ij} + ml_{ji} + mr_{ji} \leqslant (1 - v_{ij}), \quad \forall i < j,\ i,j \in N \tag{29}$$

A first-come-first-served policy with respect to the ship indices is enforced by constraint (30). This constraint makes sure that no ship $j > i$ can be placed in a lockage $l < k$ when ship i is transferred in lockage k.

$$\sum_{k<c,\ k \in M} (f_{ik} - f_{jk}) \geqslant 0, \quad \forall i < j,\ i,j \in N, c \in M \tag{30}$$

Constraints (31)–(35) formulate bounds and integrality constraints on the variables.

$$left_{ij},\ b_{ij},\ ml_{ij},\ mr_{ij} \in \{0,1\}, \quad \forall i \neq j,\ i,j \in N \tag{31}$$

$$v_{ij} \in \{0,1\}, \quad \forall i < j,\ i,j \in N \tag{32}$$

$$x_i,\ y_i \in \{0,1,\ldots,\infty\}, \quad \forall i \in N \tag{33}$$

$$z_k \in \{0,1\}, \quad \forall k \in M \tag{34}$$

$$f_{ik} \in \{0,1\}, \quad \forall i \in N, k \in M \tag{35}$$

4. Algorithms for the ship placement problem

Three algorithms have been developed for solving the ship placement problem. We first describe a top level method that is used by all three algorithms. We then present a solution method based on the MILP model discussed in Section 3.2, where the FCFS constraint has been exploited to decompose the problem, and speed up the solution process (Section 4.2). Next, we shortly describe the modified left–right–left–back heuristic (Section 4.3), followed by the introduction of a multi-order best fit heuristic for the ship placement problem (Section 4.4).

4.1. Top-level algorithm

All the ship placement algorithms here presented consider the ship list as a first-come-first-served list. This means that the first ship on the list will be presented for placement first, followed by the second ship, etc. If adding a ship results in a feasible lockage, the ship is removed from the ship list and the next ship is added to the lockage. This process is iterated until adding another ship is no longer possible. At this point, the chamber is considered full, and the algorithm returns the feasible lockage to the invoking method. This process is repeated until all ships have been placed

in a lockage. The pseudo code of the top level method is presented in Algorithm 1. The ship placement algorithms, either exact or heuristic, are called in line 6 of Algorithm 1.

Algorithm 1. Pseudo code of the top level of the ship placement algorithm

```
Require: ShipList
1: LockageList ← empty list
2: Lockage ← new Lockage
3: while not all ships placed do
4:   if Placement feasible then
5:     Lockage ← Placement
6:     add next ship of ShipList to Placement
7:   else
8:     add Lockage to LockageList
9:     Clear Placement
10:  end if
11: end while
12: return LockageList
```

Algorithm 2. Pseudo code of the left-right-left-back heuristic

```
Require: Chamber C
Require: ship to add S
1:   if C empty then
2:     C ← S at position (0,0)
3:     return C
4:   else
5:     leftPos ← lowest possible y-position at left quay
       {Position A}
6:     rightPos ← lowest possible y-position at right
       quay  {Position B}
7:     if leftPos(y) ⩽ rightPos(y) and leftPos feasible then
8:       C ← S at leftPos  {Position A}
9:       return C
10:    else if rightPos(y) < leftPos(y) and rightPos feasible then
11:      left2Pos ← lowest possible (x,y)-position right from
         a ship  {Position C}
12:      backPos ← lowest possible (x,y)-position behind
         another ship  {Position D}
13:      if backPos feasible then
14:        C ← S at backPos  {Position D}
15:        return C
16:      else if left2Pos feasible then
17:        C ← S at left2Pos  {Position C}
18:        return C
19:      else
20:        C ← S at rightPos  {Position B}
21:        return C
22:      end if
23:    else
24:      set C infeasible
25:      return C
26:    end if
27:  end if
```

4.2. Exact approach

As the calculation times for the previously introduced mathematical model turned out to be very long (see Section 5.2), a decomposition method was developed for tackling the ship placement problem. The decomposed model exploits the first-come-first-served constraint with respect to the ship indices explicitly by solving the placement problem for a single lockage at a time. Ships are added to the chamber one at a time in the order in which they appear in the ship list until no feasible solution can be found. The (optimal) last found feasible solution is then returned, and the algorithm continues with the remaining ships. Thus, the problem to be solved is always relatively small (in reality, the number of ships that can fit in a single lockage is often not more than 18). Furthermore, optimality is guaranteed, while shorter calculation times are obtained compared to solving the problem for all ships at the same time.

Most of the constraints, parameters and variables in this model are the same as those from Section 3. In the model below, only constraints that have been altered are added explicitly. For the unaltered ones, we refer to the previous model. The objective in this solution method is optional, as we only need to find a feasible solution for the current number of ships.

$$\textit{minimise} \sum_{i=1}^{N} l_i \, y_i \tag{36}$$

s.t.

$$left_{ij} + left_{ji} + b_{ij} + b_{ji} \geqslant 1, \quad \forall i < j, \; i,j \in N \tag{37}$$

(3), (4), (7), (8), (13)–(26)

$$x_i \leqslant x_j, \quad \forall i < j, \; i,j \in N : w_i = w_j, l_i = l_j \tag{38}$$

$$left_{ij}, \; b_{ij}, \; ml_{ij}, \; mr_{ij} \in \{0,1\} \quad \forall i,j \in N \tag{39}$$

$$0 \leqslant x_i \leqslant W - w_i, \quad \forall i \in N \tag{40}$$

$$dF_i \leqslant y_i \leqslant L - l_i - dB_i, \quad \forall i \in N \tag{41}$$

Constraint (37) guarantees that for each ship tuple in the lockage, one is left of and/or behind the other. This corresponds to the single chamber version of constraint (2). Constraint (38) is new, and breaks some of the symmetry that is introduced when the problem deals with ships of the same size, by imposing an order in their x-position based on their position in the ship list. Thus, if two ships i and $j > i$ are identical, x_i will never be larger than x_j, and a solution where ship i and ship j swap places will not be considered when searching for/proving the optimal solution. Adding this constraint results in a considerable speedup of the solution generation when an objective is taken into account, as opposed to returning the first feasible result.

The combination of the mooring constraints and the integer widths and lengths forces the x and y variables to be integer in any feasible solution. As a result, they can be defined as real variables, with bounds that ensure that they are placed inside the chamber's dimensions. Furthermore, we can imply bounds on the y_i variables that take the minimal distance from the front and back doors into account.

An additional speedup can be achieved by heuristics that determine a lower bound on the number of ships that can be positioned within the chamber. First, a fast heuristic places as many ships as possible in the chamber. Starting from this lower bound, more ships are added using the decomposed MILP model until optimality is reached. Using a fast and effective heuristic enables generating optimal solutions faster compared to applying the MILP model at each step.

The decomposition method can be considered a feasibility problem, which can be solved faster than the corresponding optimisation problem. There is no difference between the solutions of the feasibility problem and those of the optimisation problem apart

from the position of the ships in the chamber. These actual positions do not influence the objective function value. Nevertheless, considering an objective such as (36) may produce solutions that are more 'natural looking' or reflect a common policy. I.e. the resulting placements have some features in common with the results that are currently produced by human planners. Examples are: putting the first arriving ships at the front most positions, mooring as many ships as possible to the left quay, etc.

4.3. left-right-left-back heuristic

The second approach is a heuristic packing algorithm, namely a problem specific left-right-left-back placement algorithm, based on the approach by Verstichel and Vanden Berghe (2009). This heuristic has been adapted to produce solutions that satisfy the mooring constraints described in Section 3.1. It should be noted that the safety distance constraints are not taken into account by this heuristic. Therefore it will only be tested for the inland setting. The algorithm starts by checking the feasibility of four different positions in the chamber for the current ship. The heuristic will place the ship at the last feasible position among the four evaluated positions. The first ship in the chamber is always placed in the front most left corner. The different positions are visualised in the top row of Fig. 5 while the resulting packing is visualised in the bottom row. In Fig. 5(a), the ship can be placed at positions A, B, C and D. The left–right–left–back heuristic inserts the ship at position D. In Fig. 5(b), the ship can be placed at positions E, F and G. Inserting it at position H would violate the mooring constraint. The heuristic therefore adds the ship at position G, as this is the last evaluated position that is feasible. The only feasible positions in Fig. 5(c) are I and J as position K violates the mooring constraint. The ship is now added at position J. The left-right-left-back algorithm is called in line 6 of the top level ship placement algorithm (Algorithm 1).

4.4. Multi-order best fit heuristic

The multi-order best fit heuristic for the ship placement problem is the third approach. It is based on the three-way best fit heuristic for the orthogonal strip packing problem (Verstichel, De Causmaecker, & Vanden Berghe, 2013), which is a constructive method that obtains state-of-the-art results on a large set of benchmarks for the orthogonal strip packing problem. Due to the similarities between strip packing and ship placement, this heuristic can be transformed into a method for the ship placement problem. Other examples exist where strip packing algorithms are used to tackle similar problems (Lodi, Martello, Monaci, 2002; Lodi, Martello, Vigo, 2002). In the following paragraph we present the multi-order best fit heuristic. For this presentation, we limit the placement strategies to the 'leftmost' policy, and use the decreasing width, length and surface orderings.

The pseudo code of the multi-order best fit heuristic for the ship placement problem is added in Algorithm 3, which also contains references to the parts of Fig. 6 that are selected in that particular line of the pseudo code. In what follows, this heuristic is informally described.

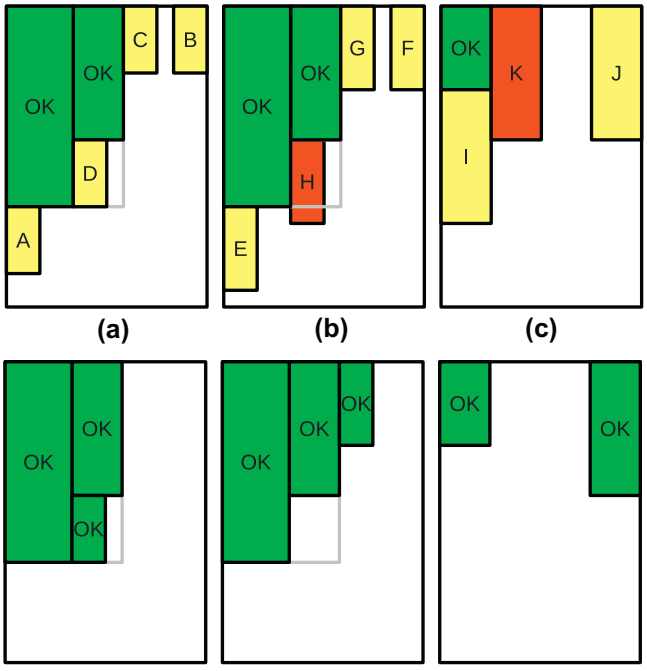

Fig. 5. A visual representation of the left-right-left-back heuristic.

Algorithm 3. Pseudo code of the multi-order best fit heuristic using the leftmost placement policy for the lock scheduling problem.

```
Require: lockage L
Require: ship to add S
1:  if L empty then
2:    L ← S at position (0,0)
3:    return L
4:  else
5:    while next ordering policy Ordering available do
6:      newL ← new Lockage
7:      ShipList ← all ships in L
8:      ShipList ← S
9:      order ships using Ordering
10:     while not all Ships added to newL do
11:       Find Lowest Gap
12:       while next Best-Fitting Ship available do
13:         if Ship can be moored to the left gap defining
    ship then
14:           newL ← Ship at leftmost position in gap
    (Subfigure a)
15:           Raise chamber skyline to reflect addition of
    Ship
16:           goto 10
17:         else if Ship can be moored to the right gap
    defining ship then
18:           newL ← Ship at rightmost position in gap
    (Subfigure b)
19:           Raise chamber skyline to reflect addition of
    Ship
20:           goto 10
21:         end if
22:         Current ship cannot be placed in the gap
    (Subfigure c)
23:       end while
24:       Raise Gap to Lowest Neighbour
25:     end while
26:     if Packing length ⩽ Chamber length then
27:       return newL
28:     end if
29:   end while
30:   set newL infeasible (No feasible packing could be
    produced)
31:   return newL
32: end if
```

Similar to the previous two algorithms, ships are added to the chamber one by one until no more ships can be added without violating constraints. The multi-order best fit heuristic orders the set of ships currently under consideration for placement in the chamber by decreasing width. In *step* (i), the heuristic detects the front most free space in the chamber, called the gap. The first ship in the ordered list with a width that is smaller than or equal to that of the gap is then selected for placement. It should be recalled that this list contains only those ships that are currently under consideration for placement in the chamber. In *step* (ii), this ship will first be placed at the left-hand side of the gap. When this leads to a feasible placement, the heuristic proceeds to *step* (i). When the left-hand side placement leads to a constraint violation (for example: the current ship is not allowed to moor to the ship at the left-hand side of the gap), the heuristic proceeds to *step* (iii), where the ship is placed at the right-hand side of the gap. Once again the feasibility of this placement is checked. In case a feasible placement is obtained, the heuristic will proceed to *step* (i). If this alternative placement also leads to an infeasible solution, the heuristic selects the next ship in the ordered list that fits the width of the gap, and proceeds to *step* (ii). If at any time during the search none of the remaining ships fit the width of the gap, the gap is filled up to the level of the least protruding gap defining ship. The search procedure then proceeds to *step* (i) and it continues until all ships are placed. At this time, the occupied chamber length is checked. If this length is smaller than or equal to the length of the chamber, the obtained (feasible) solution is returned. Otherwise, a second attempt is made at solving the problem by ordering the ships by decreasing length. Again, the constructed solution is returned if feasible. Otherwise, a final attempt is made by ordering the ships by decreasing surface. By using the alternative orderings only when the previous ones have failed to construct a feasible solution, the computational cost of using the multiple orderings is very small compared to using only one ordering. This is illustrated with a small example: The first three ships can be placed in a feasible way using the decreasing width order, so neither the decreasing length nor the decreasing surface orderings will be used. If, when adding a fourth ship, the heuristic fails to find a feasible solution, it will make a second attempt by applying the decreasing length order. When a feasible solution is found based on this ordering, the algorithm continues by adding a fifth ship. The algorithm then applies the decreasing width order and uses the decreasing length and the decreasing surface orders only when the previous one failed to generate a feasible solution. In Section 5 we show that this multi-order approach does indeed generate better results than any of the orderings on their own, while the computational cost remains very small. Changing the alternative orderings based on the problem instance at hand can be interesting. This is particularly the case when comparing orderings for the inland and port settings.

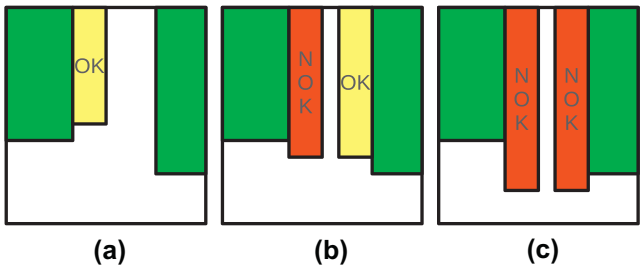

Fig. 6. Visual example of the steps that are taken when trying to place a ship in the gap. OK indicates a feasible position for the ship, NOK an infeasible one.

5. Experiments

5.1. Experimental setup

We compare the performance of the three algorithms on a set of (simulated) real-life instances. First, the exact decomposition approach is compared to a straightforward implementation of the mathematical model from Section 3.2 for a set of small test instances. Next, the effects of applying different orderings in the multi-order best fit heuristic are compared. Finally, the results of the multi-order best fit heuristic are compared with those obtained by the exact approach and the left–right–left–back heuristic.

The first test set contains examples of ship arrivals for an inland setting, with a wide variety of properties. These properties are the number of ships in the problem, the size of the chamber, the sizes of the ships and the order in which the ships need to be placed. Both the chamber sizes and ship dimensions correspond to the actual dimensions of the locks and traffic on the Albertkanaal (Belgium). The ship dimensions have been randomly sampled from all traffic on the Albertkanaal over a 12 month period. In this test set, the ships are allowed to moor to any other ship. Thus, the set $MOOR_i$ will contain all ships that are not shorter than ship i. Ship dependent safety distances are not used in the inland test set. We assume that all safety distances are already part of the ship and chamber dimensions. The test set is available online (Verstichel, 2010). Ten instances were generated for each instance size. The small instances (containing between 10 and 25 ships) were used for the experiments from Section 5.2 only. The properties of the inland instances are:

- Number of ships: 10, 20, 50, 100, 200, 500, 1000, and 10–25.
- Chamber size: 16 m $\times$ 136 m, 24 m $\times$ 200 m.
- Ship size: between 4.25 m $\times$ 16.27 m and 10.50 m $\times$ 110 m.

The port test set is extracted from one month of actual lockage operations in the Port of Antwerp. Ship size dependent safety distances must be taken into account, and two different ship types are considered. The presence of tugboats, which leave the chamber before the lockage operation, adds an extra dimension to the safety distances. No ships are allowed to moor to a 'SEA' ship, which can only moor directly to the quay, while 'BARGE' ships may moor to one another. The chambers correspond to those of the Berendrecht (BE), Zandvliet (ZV), Boudewijn (BO), Van Cauwelaert (VC), Royers (RO) and Kallo (KL) lock. The port instances can be obtained upon e-mail request to the corresponding author. Some figures about the port test instances include:

- Total number of ships: ~9000.
- Number of ships per lockage: 1–18.
- Chamber size: between 35 m $\times$ 270 m and 68 m $\times$ 500 m.
- Ship size: up to 43 m $\times$ 350 m.

All experiments were performed on a Dell Optiplex 790 with an Intel (R) Core (TM) i7-2600 (3.40 GHz) and 8 GB of memory running a 64-bit Linux Mint. Both the multi-order best fit and the left-right-left-back heuristic were implemented using Sun JDK 1.6. The optimal solutions were obtained with Gurobi 5.1 under an academic license, with a time limit of 1 h.

5.2. Exact approach

We compare the full model from Section 3.2 with the decomposed model from Section 4.2. Both solution methods always return an optimal solution to the ship placement problem. The exact position of each ship in the chambers, however, may differ

Table 1
Comparison of the average and maximum calculation time in seconds for the full and decomposed models. Ten instances were solved for each problem size.

#Ships	Average time (s)		Maximum time (s)	
	Full	Decomp	Full	Decomp
10	8.55	0.44	30.01	1.29
11	25.30	0.31	121.26	0.73
12	61.80	0.52	393.33	2.19
13	167.32	1.53	988.74	11.83

between the two approaches. Both models were first evaluated on a set of inland instances with 10–13 ships, using the large chamber (24 m $\times$ 200 m). For larger instances, the original model frequently required more than 1 h to attest optimality. Table 1 shows that the decomposed model exploiting the FCFS constraints is significantly faster than the original model. This indicates that generating a feasible solution to many single-lockage ship placement problems is easier than solving a single multi-lockage instance to optimality. Consider, for example, a 13 ship instance from Table 1 with a known upper bound of 5 required lockages. The experiments show that the corresponding 13 single-lockage ship placement problems are solved much faster than the single 5 lockage instance. In the second experiment, we compare the time required by the full model to discover an optimal solution, and the time require to prove its optimality. This is achieved by providing the full model with a lower bound, generated from the optimal solutions of the decomposed model. Adding this lower bound drastically reduced the computation time and enabled solving instances with up to 21 ships in less than 1 h. For instances with 22 and more ships, optimality was often not reached in less than 1 h, and for sizes 25 and above the full model frequently failed to find an initial feasible solution in the same amount of time. Fig. 7 shows a summary of this experiment, plotting the computation time of the full model with and without attesting optimality, and the decomposed approach.

All other experiments are based on the decomposed model only. Fig. 8 shows the solution of one specific ship placement problem, where 10 ships are placed in one single chamber. This solution was found in 1 s and leaves less than 10% of the chamber's surface free.

5.3. Inland setting

5.3.1. Multi-order best fit with different orderings

In this part of the experiments, we apply the multi-order best fit heuristic to the test instances, and compare the results of a single ordering with those of applying three orderings (Section 4.4). The single-ordering heuristics are stripped versions of the multi-order best fit heuristic, based on a single ship ordering. The results over all inland test instances for the large chamber are shown in Table 2, which also contains the total number of occupied chambers and the total calculation time needed for solving the entire test set. These results clearly indicate that the combination of multiple orderings outperforms the heuristics applying any of these orderings on their own. The calculation time needed by the multi-order best fit heuristic stays well below the combined calculation times of the single-ordering best fit heuristics. This is due to the conditional application of the alternative orderings, i.e. only when another ordering fails. The results for the small chamber have been omitted because the heuristics generated a different number of lockages for a few instances only.

From these experiments, we conclude that applying alternative orderings when necessary, leads to an improved solution quality compared to applying one single ordering. Furthermore, this quality increase comes at a low computational cost. Finally, we point out that the multi-order best fit heuristic is significantly better than using the best of the single-ordering heuristics. I.e. conditionally applying the alternative ordering strongly outperforms selecting the best of the three single-ordering heuristics (p-value < 10^{-4}).

5.3.2. Results for the ship placement algorithms

This series of experiments compares three approaches to the inland setting:

- Decomposed MILP approach (Section 4.2),
- left-right-left-back problem specific heuristic (Section 4.3),
- Multi-order best fit heuristic (Section 4.4).

We performed the experiments on all inland test instances and the results are presented in Table 3.

The multi-order best fit heuristic always outperforms the left-right-left-back heuristic for the instances with more than 50 ships and for 47% of the smaller test instances. The solution quality of both approaches is equal for the remaining smaller test instances. The results of the best fit heuristic are statistically better than those of the left-right-left-back heuristic with an almost 100% certainty, based on a pairwise T-test applied to the entire set of test results. Furthermore, optimality with respect to the required number of chambers is reached for 35 out of 70 instances, while the multi-order best fit heuristic requires only a fraction of the time needed by the MILP approach. When considering the total computation time over all instances, the best fit heuristic consumes only

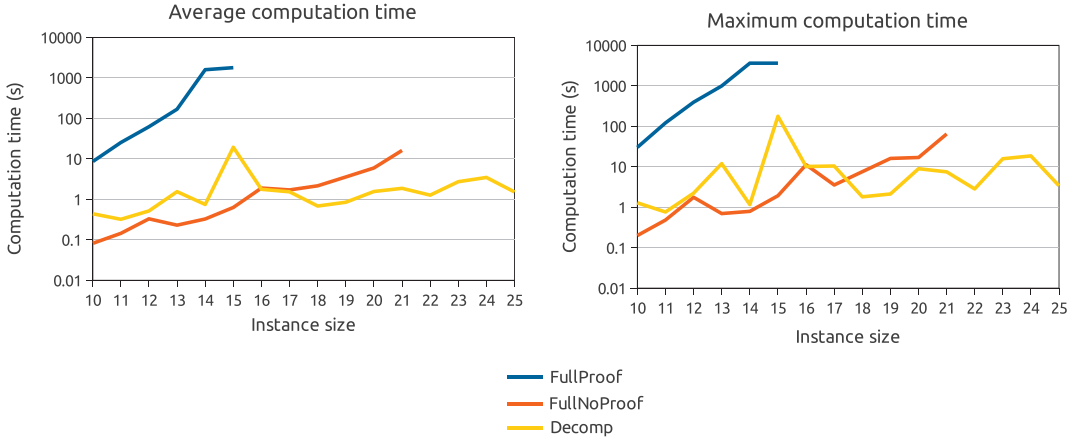

Fig. 7. Comparison of the computation time required by the different exact approaches. 'FullNoProof' shows the time required by the full model to find the optimal solution, while 'FullProof' also includes the time required to prove optimality. 'Decomp' shows the time required by the decomposition approach to find and prove the optimal solution.

Fig. 8. Example of an instance with 10 ships where less than 10% of the chamber's surface is still free.

Table 2
Results of the multi-order best fit heuristic with different orderings and the large chamber. Results are averaged over 10 instances for each problem size. Best results are presented in bold. BF denotes best fit, followed by the applied ordering. # Denotes the average number of lockages required over 10 instances of the given size.

	#Ships	BF width		BF length		BF surface		Multi-order BF	
		#	time (s)	#	time (s)	#	time (s)	#	time (s)
	10	**2.1**	0.001	**2.1**	0.001	**2.1**	0.002	**2.1**	0.001
	20	**3.9**	0.001	4	0.001	**3.9**	0.001	**3.9**	0.001
	50	9.4	0.001	9.4	0.002	**9.3**	0.001	**9.3**	0.002
	100	18.5	0.002	18.5	0.003	18.2	0.002	**18.1**	0.003
	200	36.9	0.005	37	0.005	36.8	0.005	**36.3**	0.006
	500	91.2	0.010	91.7	0.010	90.6	0.009	**89.6**	0.014
	1000	179.3	0.020	180	0.020	178.8	0.021	**176.6**	0.029
∑	18800	3413	0.405	3427	0.422	3397	0.410	**3359**	0.552

Table 3
Results of the different ship placement algorithms, computed for the large chamber. All results are averaged over 10 instances for each problem size.

#Ships	MILP		LRLB			Multi-order best fit		
	#	Time (s)	#	%Gap	Time (s)	#	%Gap	Time (s)
10	2.1	0.449	2.3	10.00	0.001	2.1	0.00	0.001
20	3.8	1.553	4.2	12.50	0.001	3.9	3.33	0.001
50	9.2	6.950	10.3	12.00	0.001	9.3	1.11	0.002
100	17.6	53.368	20.3	15.49	0.001	18.1	2.85	0.003
200	35.0	24.328	40.4	15.49	0.003	36.3	3.68	0.006
500	87.0	49.300	99.4	14.28	0.004	89.6	3.00	0.014
1000	171.4	114.335	196.6	14.73	0.009	176.6	3.04	0.029

Table 4
Comparison of the average performance of the increasing arrival time order best fit heuristic (Arrival BF), multi-order best fit heuristic (Multi-Order BF) and decomposed MILP approach (Exact) when ships are queueing at the lock. The solution properties are average number of ships placed (#), average computation time (Time) and the number of improved instances (Impr) with the total number of instances for each lock between parentheses.

Lock	Arrival BF		Multi-Order BF			Exact		
	#	Time (ms)	#	Impr	Time (ms)	#	Impr	Time (s)
BO	10	0.5	11.8	5(6)	1.0	12.7[a]	4(6)	1070.4[a]
VC	8.6	2.6	9.5	6(8)	4.6	10.1	4(8)	165.4
RO	6.1	0.3	6.7	3(9)	0.5	7.3	6(9)	10.5
KL	7.6	0.7	8.5	5(11)	1.6	9.5	9(11)	3.8

[a] The time limit of 1 h was reached on one instance.

0.0002% of the MILP's computation time. The computation times for the individual instances reveal that, on average, the best fit heuristic consumes 0.5% of the time required by the MILP approach. For the smaller instances (<200 ships), the difference in number of chambers between the optimal solution and the best fit heuristic was at most 1 on a total of 4. For the larger ($\geqslant$200 ships) instances this difference was never more than 7 chambers out of 173 or 4%, while the average optimality gap was 3.24%.

When looking at the left-right-left-back heuristic, the difference with the optimal solution for the smaller instances is at most 5 on a total of 16. For the larger instances, the difference increases to a maximum of 29 chambers out of 173 or 16.7%, which is over four times the optimality gap obtained by the best fit heuristic.

The number of chambers needed by the MILP approach and the multi-order best fit heuristic were equal for all instances when using the small chamber. The left-right-left-back heuristic, however, needed additional chambers for 39 out of 70 instances, while its average optimality gap was 1.11%. Due to these small differences, the results for this test setting were omitted from the paper.

5.4. Port setting

The following three approaches are compared for the port setting:

- Increasing arrival time order best fit heuristic,
- Multi-order best fit heuristic,
- MILP approach.

More specifically, we introduce the increasing arrival time ordering, and compare its stand alone performance to that of the multi-order best fit using four orderings (arrival, width, length, surface). Both approaches are also compared to the exact decomposition approach with a time limit of 1 h per iteration. An increasing arrival time ordering may be preferred in port environments, as this ordering places the first arriving ships at the front most positions in the chamber. This does however correspond to a single ordering heuristic, which may lead to significantly worse results than using a combination of orderings. The objective is to place as many ships as possible in the chamber under a first-come-first served restriction. These instances were selected from the port data where a queue of ships was waiting to be transferred by the lock, and where the optimal solution contained at least one ship more than the increasing arrival solution. The results from Table 4 show that ignoring the arrival order of the ships when selecting their position leads to a significantly higher utilisation of the chamber. The exact approach is frequently able to find better solutions than the heuristics, but requires significantly more computation time. Setting a lower time limit for the exact approach may bring the computation times closer together, but also increases the risk of missing the optimal solution. Fig. 9 shows an example of how the usage of the multi-order best fit heuristic outperforms the stand-alone increasing arrival ordering.

The second part of the port experiments compares the algorithmic results with the manual solutions generated by the lock masters. Using the available data, we can automatically place the set of ships processed in a manually obtained lockage. If not all the ships from this set can be placed, we consider it an algorithmic failure. Table 5 shows the number of manual lockages that could be reconstructed by each solution method, and the average computation time needed. The results show that the multi-order best fit heuristic is able to reconstruct 99.3% of the solutions, while the exact approach reconstructs all lockages. The exact solutions do however come at a computational cost. While the heuristic solves each in-

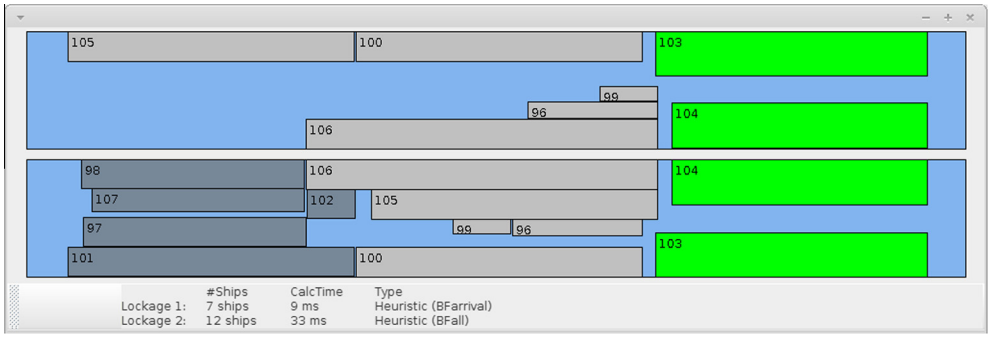

Fig. 9. Example of an instance for which the multi-order best fit (lower solution) strongly outperforms a single increasing arrival ordering best fit (upper solution).

Table 5
Comparison of the increasing arrival time order best fit heuristic (Arrival BF), multi-order best fit heuristic (Multi-Order BF) and exact MILP approach (Exact) when reproducing real-life lockages. # Denotes the number of lockages that could be reproduced, with the total number of lockages between parentheses, and Time shows the average computation time.

Lock	Arrival BF		Multi-Order BF		Exact	
	#	Time (ms)	#	Time (ms)	#	Time (ms)
BE	357(362)	0.22	360(362)	0.29	362(362)	11.07
ZV	390(390)	0.08	390(390)	0.08	390(390)	5.89
BO	397(405)	0.07	401(405)	0.07	405(405)	55.68
VC	343(354)	0.08	344(354)	0.07	354(354)	7.54
KL	545(546)	0.06	546(546)	0.06	546(546)	3.24
RO	608(614)	0.05	611(614)	0.04	614(614)	2.93
Total	98.84%	0.09	99.29%	0.10	100.00%	14.39

stance in less than 20 ms, the exact approach requires up to 14 s for a single instance. The single-order best fit heuristics performs only slightly worse than the other approaches, managing to reconstruct 98.8% of the instances in 0.1 ms on average.

6. Conclusion

Locks are a key component in a port's infrastructure and an essential part on many waterways. Efficient lockage operation becomes increasingly important due to the growing number of goods transported by ships. We presented the ship placement problem, which is an important part of the lock scheduling problem. It has been identified as a variant of the 2D bin packing problem with additional constraints, and has proven to be different from 2D bin packing. A mathematical model for the ship placement problem was developed, and a fast exact decomposition algorithm for the ship placement problem was created by exploiting the precedence constraints of the lock scheduling problem. Problem instances with up to 1000 ships can be solved to optimality in less than 500 s by the general purpose solver Gurobi 5.1. These relatively long computation times, which sometimes differ strongly among similar instances, render this exact approach less than ideal for real-life usage. As the ship placement problem needs to be solved many times while searching for a good solution for the entire lock scheduling problem, short and stable calculation times are required. We have therefore introduced the multi-order best fit heuristic, which is based on an excellent heuristic for the orthogonal stock-cutting problem (Verstichel et al., 2013). When the current ordering fails to produce a good solution, the multi-order best fit heuristic applies an alternative ordering in search for a feasible solution. Thus, a significant performance increase over each individual ordering is gained, while only requiring very little additional calculation time. We then compared the multi-order best fit heuristic to a MILP approach and to the existing problem specific left-right-left-back heuristic for the ship placement problem on a large test set. The experiments showed that the multi-order best fit heuristic is significantly better than the left-right-left-back heuristic. It generates solutions for large instances with up to 1000 ships in less than 0.1 s with an average optimality gap of 3.24%. The solution methods were also applied to real-life instances from a port, on which they performed excellently. The multi-order best fit heuristic was able to reconstruct 99.3% of the lockages performed by the lock masters, while never requiring more then 6 ms of calculation time. The exact approach reconstructed all lockages, but did require up to 89 s to do so. Due to this combination of high solution quality and low calculation times, the multi-order best fit heuristic is an ideal solution method for the ship placement problem. Human experts who currently solve the ship placement problem in real-life situations often group ships of similar size together into one larger block in order to simplify the problem. An additional benefit of the multi-order best fit heuristic is that it tends to group ships of similar size. Consequently, the result is easier for the human experts to analyse while increasing the possibility that they will agree to the quality of the presented solution. Live-tests of the algorithms confirmed both their real-life applicability and the benefits of the aforementioned best fit heuristic's grouping behaviour.

Acknowledgements

Research funded by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

We would like to thank the Scheepvaartmanagement of the Port of Antwerp for sharing their experience and real-life data on the ship placement problem. The real-life data provided by IT-Bizz and nv De Scheepvaart was also greatly appreciated.

References

Bish, E. (2003). A multiple-crane-constrained scheduling problem in a container terminal. *European Journal of Operational Research, 144*, 83–107.

Chen, L., Bostel, N., Dejax, P., Cai, J., & Xi, L. (2006). A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. *European Journal of Operational Research, 181*, 40–58.

Coene, S., Spieksma, F., & Vanden Berghe, G. (2011). The Lockmaster's problem. Technical Report KBI-1126. KU Leuven. <https://lirias.kuleuven.be/handle/123456789/318524>.

Cullinane, K., & Khanna, M. (2000). Economies of scale in large containerships: Optimal size and geographical implications. *Journal of Transport Geography, 8*, 181–195.

European Commission (2009). Communication from the commission: A sustainable future for transport-towards an integrated, technology-led and user friendly systems. <http://ec.europa.eu/transport/media/publications/doc/2009_future_of_transport_en.pdf>.

European Commission (2011). Transport White Paper: Roadmap to a single European transport area towards a competitive and resource efficient transport system. <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2011:0144:FIN:EN:PDF>.

Günther, H. O., & Kim, K. H. (2006). Container terminals and terminal operations. *OR Spectrum, 28*, 437–445.

Leung, J. Y-T., Tam, T. W., Wong, C. S., Young, G. H., & Chin, F. Y. L. (1990). Packing squares into a square. *Journal of Parallel and Distributed Computing, 10*, 271–275.

Lodi, A., Martello, S., & Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research, 141*, 241–252.

Lodi, A., Martello, S., & Vigo, D. (2002). Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics, 123*, 379–396.

Nauss, R. M. (2008). Optimal sequencing in the presence of setup times for tow/barge traffic through a river lock. *European Journal of Operational Research, 187*, 1268–1281.

Notteboom, T. E., & Rodrigue, J. (2005). Port regionalization: Towards a new phase in port development. *Maritime Policy & Management, 32*, 297–313.

Pisinger, D., & Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization, 2*, 154–167.

Smith, L., Nauss, R. M., Mattfeld, D. C., Li, J., Ehmke, J. F., & Reindl, M. (2011). Scheduling operations at system choke points with sequence-dependent delays and processing times. *Transportation Research Part E: Logistics and Transportation Review, 47*, 669–680.

Smith, L. D., Sweeney, D. C., II, & Campbell, J. F. (2009). Simulation of alternative approaches to relieving congestion at locks in a river transportation system. *Journal of the Operational Research Society, 60*, 519–533.

Stahlbock, R., & Voß, S. (2008). Operations research at container terminals: A literature update. *OR Spectrum, 30*, 1–52.

Verstichel, J., & Vanden Berghe, G. (2009). A late acceptance algorithm for the lock scheduling problem. *Logistik Management*, 457–478.

Verstichel, J. (2010). Online instances for the ship placement problem. <http://allserv.kahosl.be/~jannes/lockplanning/index.html>.

Verstichel, J., De Causmaecker, P., & Vanden Berghe, G. (2011). Scheduling algorithms for the lock scheduling problem. *Procedia-Social and Behavioral Sciences, 20*, 806–815.

Verstichel, J., De Causmaecker, P., Vanden Berghe, G. (2013). An improved best fit heuristic for the orthogonal strip packing problem. *International Transactions in Operational Research*, http://dx.doi.org/10.1111/itor.12030.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research, 183*, 1109–1130.

Wiese, J., Suhl, L., & Kliewer, N. (2010). Mathematical models and solution methods for optimal container terminal yard layouts. *OR Spectrum, 32*, 427–452.

Wilson, H. G. (1978). On the applicability of queueing theory to lock capacity analysis. *Transportation Research, 12*, 175–180.