Contents lists available at ScienceDirect

Transportation Research Part E

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

Improved formulations and an Adaptive Large Neighborhood Search heuristic for the integrated berth allocation and quay crane assignment problem

Çağatay Iris [a,b,*], Dario Pacino [a], Stefan Ropke [a]

[a] Department of Management Engineering, Technical University of Denmark, 2800 Copenhagen, Denmark
[b] School of Civil and Environmental Engineering, Nanyang Technological University, 639798 Singapore, Singapore

A R T I C L E I N F O

Article history:
Received 13 November 2016
Received in revised form 17 May 2017
Accepted 30 June 2017

A B S T R A C T

This paper focuses on the integrated berth allocation and quay crane assignment problem in container terminals. We consider the decrease in the marginal productivity of quay cranes and the increase in handling time due to deviation from the desired position. We consider a continuous berth, discretized in small equal-sized sections. A number of enhancements over the state-of-the-art formulation and an Adaptive Large Neighborhood Search (ALNS) heuristic are presented. Computational results reveal that the enhancements improve many of the best-known bounds, and the ALNS outperforms the state-of-the-art heuristics for many instances. We also conduct further analysis on a new larger benchmark.

1. Introduction

Recent statistics show that world container port throughput is increased by an estimated 5.1% to 684.1 million Twenty-foot Equivalent Units (TEUs) in 2014 (UNCTAD, 2015). The same report illustrates that large container terminals can process more than 30 million containers a year. These high volumes have increased the need for more efficient container terminal management. Due to the growing number of terminal operations and the interconnection between them, the complexity of planning has also increased tremendously. In this respect, the use of operations research techniques has caught the attention of both the industry and academia (see Stahlbock and Voß (2007) for a review of container terminal problems and operations research).

The productivity of a container terminal heavily relies on the efficient use of its resources. Focusing on quayside operations, Quay Crane (QC) management, and the usage of the berthing area are among some of the most important container terminal planning problems. In the Berth Allocation Problem (BAP), it is decided when and where vessels should be berthed. In the QC assignment problem, the number of QCs to be assigned to each ship is decided.

A key input to the BAP is the handling time of each vessel. This time depends on the number of QCs assigned to the vessel, a decision that belongs to the QC assignment problem. The feasibility of a QC assignment, on the other hand, depends on the berthing time of the vessels, a decision that belongs to the BAP. Thus, the two problems are closely connected, and it is, therefore, likely that higher quality solutions can be obtained by solving a model that integrates the two problems.

The literature distinguishes between two such problems. The Berth Allocation and Quay Crane Assignment Problem (BACAP) where a decision is made on the number of QC to assign (e.g. Meisel and Bierwirth, 2009; Iris et al., 2015); and

* Corresponding author at: Department of Management Engineering, Technical University of Denmark, 2800 Copenhagen, Denmark.
E-mail address: cagai@dtu.dk (Ç. Iris).

http://dx.doi.org/10.1016/j.tre.2017.06.013
1366-5545/© 2017 Elsevier Ltd. All rights reserved.

the Berth Allocation and Quay Crane Assignment (Specific) and Scheduling Problem (BACASP) where specific QC scheduling is also part of the decision (e.g. Liu et al., 2006; Imai et al., 2008; Turkogullari et al., 2014).

In this paper, we consider the BACAP as introduced in Meisel and Bierwirth (2009) where the decision is over the berthing position, start and end time, and the number of QCs assigned to each vessel. The number of assigned QCs is allowed to vary within vessel's berthing time (time-variant BACAP). A marginal productivity loss is considered for QCs due to interference. Additionally, when a vessel's berthing position diverges from the desired one, the number of QC hours to service the vessel increases. This increase is caused by additional housekeeping operations or by the extra travel time needed to reach containers that were otherwise positioned near the preferred berthing position. A solution to the BACAP can be integrated into the BACASP. Park and Kim (2003) show that the specific QC assignment problem, with a pre-determined BACAP, corresponds to the shortest path problem. Recently, Turkogullari et al. (2016) presented a more unified exact approach in which cutting planes are generated online over the BACAP solutions and added to a relaxed problem. These studies further motivate the need for efficient BACAP solution methods in order to solve more complicated problems.

The BACAP naturally lends itself towards a description in a two-dimensional space. One dimension is spatial (i.e. the quay partition), which could be discrete (Cordeau et al., 2005; Buhrkal et al., 2011), continuous (Imai et al., 2005; Lee et al., 2010), or hybrid (Kordic et al., 2016). The other dimension is temporal (i.e. the planning horizon), which could be static (Park and Kim, 2003), dynamic (Imai et al., 2001; Imai et al., 2008), or cyclic (Jin et al., 2015; Imai et al., 2014). Fig. 1 shows this representation where the vertical axis represents the spatial dimension and the horizontal axis represents the temporal dimension. In this representation, a vessel is a rectangle whose time dimension depends on the number of assigned QCs at each unit of time. In this paper, the spatial dimension is considered to be a continuous berth, where we assume that a vessel can berth at any 10-meter point along the quay (i.e. the quay is discretized by 10-meter sections). The temporal property is dynamic because the arrival time of each vessel imposes a bound for the berthing start time. We assume that all parameters are known in advance (see Golias et al. (2014) and Shang et al. (2016) for berth planning problems under uncertainty).

The main contributions of this paper are twofold. First, we introduce novel valid inequalities and variable fixing methods that enhance the state-of-the-art compact model (Meisel and Bierwirth, 2009). These enhancements can also be used to improve different formulations of the BAP, BACAP or BACASP. We also discuss which state-of-the-art models can be improved with the use of each enhancement. Secondly, we present, to the best of the authors' knowledge, the first Adaptive Large Neighborhood Search (ALNS) algorithm for the BACAP. ALNS was first proposed by Ropke and Pisinger (2006), (see also Pisinger and Ropke, 2007), and has been successfully used to solve a variety of complex optimization problems (see Coelho et al., 2012; Muller et al., 2012, etc.).

Computational results reveal that the performance of the model from Meisel and Bierwirth (2009) has been significantly increased with the addition of our inequalities. The results also present improvements on the bounds published in Iris et al. (2015). The performance of each individual family of inequalities is also discussed. The proposed ALNS outperforms the state-of-the-art heuristics of Meisel and Bierwirth (2009) and Shang et al. (2016) with respect to solution quality for many instances. New best upper and lower bounds are found for all instances. The performances of the improved formulation and the ALNS are also tested against a new, and larger, benchmark.

The remainder of this paper is organized as follows. In Section 2, we present a brief literature review for the BACAP and its variants. In Section 3, we present a formal definition of the BACAP. The mathematical model proposed by Meisel and Bierwirth (2009), valid inequalities and variable fixing methods for this formulation are described in Section 4. In Section 5, we present an ALNS for the BACAP. Computational results are discussed in Section 6. Finally, we present conclusions and future research directions in the last section.

2. Literature review

The BACAP has attracted many researchers in the field. Extensive reviews on BACAP and BACASP literature can be found in Bierwirth and Meisel (2015), Carlo et al. (2015) and Iris et al. (2015). In the last, the literature is clustered according to problem structure, characteristics of the objective function and solution methods. Thus, the literature review in this paper only addresses works that have substantially contributed to the state-of-the-art, and some recent papers including significant findings for these problems are included.

One of the first works on the BACAP is presented in Park and Kim (2003). The authors solve the BACAP in a first stage with a subgradient optimization strategy. The results are then used in a second stage to solve the BACASP with dynamic programming. Meisel and Bierwirth (2009) present a different BACAP model. The authors propose various heuristics (tabu search (TS), first-come-first-served (FCFS) heuristics, and squeaky wheel optimization (SWO)) to solve this problem. Another paper, presented by Liang et al. (2009), determines the berthing positions, berthing start/end times and number of QCs to serve each vessel. They assume a time-invariant QC allocation policy (i.e. the number of QC operating on a vessel does not change in time) is applied. They solve the problem with a hybrid genetic algorithm. Giallombardo et al. (2010) have focused on a variant of BACAP called the tactical berth allocation problem (TBAP). This problem uses the concept of QC profiles. A QC profile includes the number of QCs assigned to a vessel during the time the vessel is at the port. This profile also holds some real-world requirements imposed by some terminals such as; QC movements only at shift changes, vessel priorities in terms of the number of QCs, etc. The authors solve the problem via a two-stage heuristic algorithm. Vacca et al. (2013) focus on the formulation of Giallombardo et al. (2010) and they propose the first exact decomposition framework for this variant of the

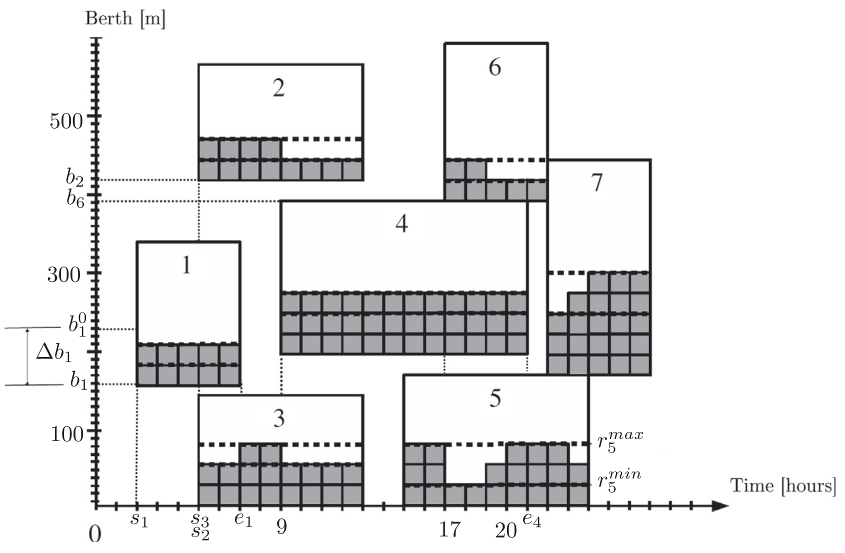

Fig. 1. Example of BACAP (Meisel and Bierwirth, 2009).

BACAP. Lalla-Ruiz et al. (2014) also focus on this TBAP formulation and implement a genetic algorithm to solve the problem. Studies that follow the QC profile approach do not solve the optimum QC assignment problem presented in this paper, those studies mainly generate a representative set of QC profiles and select one of them for each vessel. Finally, Iris et al. (2015) formulate various set partitioning formulations and column reduction techniques for the BACAP and its variants (time-variant/invariant QC allocation, finer/hybrid berth discretization, etc.).

Liu et al. (2006) is one of the first papers that focus on a variant of BACASP. The authors assume that the berthing position for each vessel is known in advance (i.e. one important component of the BAP is given as a parameter) and they determine vessel berthing start/end times, the number of QCs to assign to each vessel and which specific QC should be used. They propose two greedy heuristics to solve this problem. Imai et al. (2008) present a novel mathematical model and the first complete BACASP definition. They consider a discrete berth partition where every vessel can fit in exactly one berth. The authors propose a genetic algorithm to solve the problem. Ursavas (2014) also focuses on a BACASP with discrete berths. The author considers time-variant QC allocation, and proposes a bi-objective mathematical model. Turkogullari et al. (2014) formulate a model for the time-invariant BACASP. The authors emphazise that for large scale instances the model is not effective. Hence, they propose a post-processing cutting plane algorithm applied to the results of a BACAP solution. Experiments show that this method can solve the largest instances to optimality. Recently, Turkogullari et al. (2016) relax the assumption of time-invariant QC allocation and solve the time-variant version of the problem with an exact method. One major difference between the BACAP component of these two studies and ours is that these studies do not consider the increase in the QC capacity demand due to deviation from the desired berthing position. Rodriguez-Molins et al. (2014) have focused on a BACASP with both time-variant/invariant QC allocation. The time-variant QC version is designed to assign the QCs for specific holds. The authors have proposed a Greedy Randomized Adaptive Search Procedure (GRASP) heuristic and showed that the algorithm outperforms traditional heuristics like FCFS, etc. Recently, Li et al. (2015) have focused on a BACASP in which QC coverage ranges are also considered. The authors present a novel mathematical model which has many BACAP considerations from Meisel and Bierwirth (2009) model. They also propose a heuristic algorithm based on spatio-temporal conflicts analysis.

3. Problem description

The BACAP studied in this paper aims at finding a berthing start time and position for each vessel in the planning horizon. Moreover, the berthing end time is assumed to be a function of the number of assigned QCs. We consider a continuous berth and discretize time in units of one hour.

Fig. 1 shows an example BACAP solution in a time/quay diagram. In this example, seven vessels are berthed. Each vessel is represented by a rectangle showing the time and space occupied by the vessel. Within each rectangle, at every given time unit, each small gray square indicates one assigned QC. Every berthed vessel has an upper and lower limit on the number of assignable QCs (r_i^{min}, r_i^{max}) which are presented with bold dashed lines. These values are determined by contracts between vessel operators and the terminal, and by the size of the ship. A limited number of QCs are available at the terminal, and this determines the maximum number of QCs that can be assigned at any point in time. Every vessel has also a berthing start (s_i) and end time (e_i) on the horizontal axis. The vertical axis includes the berthing position of each vessel (b_i), and each vessel has a desired berthing position parameter (b_i^0). The deviation from desired berthing position is Δb_i. The example is explained in the next section.

There are two considerations about terminal resource utilization in Meisel and Bierwirth (2009). The authors argue that the assigned number of QC for each vessel cannot be completely used due to the interference between QCs. This means when q QCs are assigned to a vessel, the productivity is q^α QC-hours, where α is an exponent of interference ($0 \leqslant \alpha \leq 1$). Another considered aspect is the increase in the horizontal transportation when a vessel is not berthed at its desired berthing position. Meaning that there is an increase in the QC hours needed to carry out the work (i.e. fulfill the QC capacity requirement) on a vessel. This increase is proportional to the distance between the actual and the desired berthing position (Δb_i).

Meisel and Bierwirth (2009) consider a combination of time-dependent costs and QC assignment costs for the objective function. The time dependent costs are related to the berthing start and end time, while the QC dependent cost is a function of how many QCs are assigned to a given vessel. The model includes three time-dependent cost components: speed-up cost, delay cost and penalty cost. It is assumed that vessels can speed-up in the open sea and thus berth earlier than the Expected Time of Arrival (ETA). For each time unit earlier than ETA, a speed-up cost must be paid. It is, however, assumed that a fixed Earliest Start Time (EST) for each vessel exists. The problem does not allow any berthing start time before EST, since it is not feasible with respect to the maximum speed of a vessel. The problem also penalizes lateness for each vessel. If a vessel finishes its operations and departs (berthing end time) after the Expected Finishing Time (EFT), a lateness cost for each late time period must be paid. Moreover, if the berthing end time is later than the Latest Finishing Time (LFT), a one-time penalty cost is added to the cost function. There exists a trade-off between the handling time (which is also a function of the berthing position) of a vessel and the corresponding number of QCs assigned to the vessel. Finally, the cost of operating all assigned QCs of a vessel is also added to the model.

4. Mathematical model and enhancements

A complete list of notations, i.e. parameters, decision variables of the model, can be found in Table 1.

Let us now introduce the mathematical model by Meisel and Bierwirth (2009):

Table 1
BACAP mathematical notation.

Parameters and sets:	
V	Set of vessels to be served, $V \in \{1, 2, \ldots, N\}$, where N is the number of vessels to berth
T	Set of time periods (1 h), $T \in \{0, 1, \ldots, H-1\}$, where H is the planning horizon
L	Length of the quay given as a number of 10-meter sections
l_i	Length of vessel $i \in V$ given as a number of 10-meter sections
b_i^0	Desired berthing position of vessel $i \in V$
m_i	Quay crane capacity demand of vessel $i \in V$ (i.e. total QC-hours needed)
r_i^{min}	Minimum number of QCs agreed to serve vessel $i \in V$ simultaneously
r_i^{max}	Maximum number of QCs allowed to serve vessel $i \in V$ simultaneously
R_i	Set of number of QCs assignable to vessel $i \in V$, where $R_i \in \{r_i^{min}, \ldots, r_i^{max}\}$
ETA_i	Expected time of arrival of vessel $i \in V$
EST_i	Earliest starting time of vessel $i \in V$
EFT_i	Expected finishing time of vessel $i \in V$
LFT_i	Latest finishing time of vessel $i \in V$
c_i^1	Speed-up cost of vessel $i \in V$ on its journey to catch a berthing time earlier than ETA_i
c_i^2	Cost of exceeding the expected finishing time EFT_i for vessel $i \in V$
c_i^3	Penalty cost by exceeding LFT_i for vessel $i \in V$
c^4	Cost rate per QC-hour of operations
α	Interference exponent for the QCs. Only q^α effective QC hours are obtained when assigning q QCs to a ship for one hour
β	Coefficient of increase in the QC capacity demand with deviation from desired berthing position. A vessel i placed at position b_i needs $(1 + \lvert b_i^0 - b_i \rvert \beta) m_i$ effective QC hours
M	A large positive number
Q	Available number of QCs
Decision variables:	
$b_i \in \mathbb{Z}^+$	Berthing position of vessel $i \in V$
$s_i \in \mathbb{Z}^+$	Berthing start time of vessel $i \in V$
$e_i \in \mathbb{Z}^+$	Berthing end time (time when the handling ends) of vessel $i \in V$
$r_{it} \in \mathbb{B}$	1; if vessel $i \in V$ is at berth being processed in period t, 0 otherwise
$r_{itq} \in \mathbb{B}$	1; if there is exactly q QCs assigned to vessel i in period t, 0 otherwise
$\Delta b_i \in \mathbb{Z}^+$	Deviation from desired berthing position when vessel i is in position b_i, $\Delta b_i = \lvert b_i^0 - b_i \rvert$
$\Delta ETA_i \in \mathbb{Z}^+$	Earliness of vessel i to reach start-time s_i, where $\Delta ETA_i = (ETA_i - s_i)^+$
$\Delta EFT_i \in \mathbb{Z}^+$	Tardiness of vessel $i \in V$ when operations are finished later than expected finishing time, $\Delta EFT_i = (e_i - EFT_i)^+$
$u_i \in \mathbb{B}$	1; if finishing time of vessel $i \in V$ exceed latest finishing time, 0 otherwise
$y_{ij} \in \mathbb{B}$	1; if vessel $i \in V$ is berthed below vessel $j \in V$ in berth area, i.e. $b_i + l_i \leqslant b_j$, 0 otherwise
$z_{ij} \in \mathbb{B}$	1; if handling of vessel $i \in V$ ends no later than handling of vessel $j \in V$ starts in berth area, 0 otherwise

$$\min \sum_{i \in V} (c_i^1 \Delta ETA_i + c_i^2 \Delta EFT_i + c_i^3 u_i + c^4 \sum_{t \in T} \sum_{q \in R_i} r_{itq} q) \tag{1}$$

subject to

$$\sum_{t \in T} \sum_{q \in R_i} q^\alpha r_{itq} \geqslant (1 + \Delta b_i \beta) m_i \quad \forall i \in V \tag{2}$$

$$\sum_{i \in V} \sum_{q \in R_i} q r_{itq} \leqslant Q \quad \forall t \in T \tag{3}$$

$$\sum_{q \in R_i} r_{itq} = r_{it} \quad \forall i \in V, \forall t \in T \tag{4}$$

$$\sum_{t \in T} r_{it} = e_i - s_i \quad \forall i \in V \tag{5}$$

$$(t+1) r_{it} \leqslant e_i \quad \forall i \in V, \forall t \in T \tag{6}$$

$$r_{it} t + H(1 - r_{it}) \geqslant s_i \quad \forall i \in V, \forall t \in T \tag{7}$$

$$\Delta b_i \geqslant b_i - b_i^0 \quad \forall i \in V \tag{8}$$

$$\Delta b_i \geqslant b_i^0 - b_i \quad \forall i \in V \tag{9}$$

$$\Delta ETA_i \geqslant ETA_i - s_i \quad \forall i \in V \tag{10}$$

$$\Delta EFT_i \geqslant e_i - EFT_i \quad \forall i \in V \tag{11}$$

$$M u_i \geqslant e_i - LFT_i \quad \forall i \in V \tag{12}$$

$$b_j + M(1 - y_{ij}) \geqslant b_i + l_i \quad \forall i, j \in V, \quad i \neq j \tag{13}$$

$$s_j + M(1 - z_{ij}) \geqslant e_i \quad \forall i, j \in V, \quad i \neq j \tag{14}$$

$$y_{ij} + y_{ji} + z_{ij} + z_{ji} \geqslant 1 \quad \forall i, j \in V, \quad i \neq j \tag{15}$$

$$s_i, e_i \in \{EST_i, \dots, H\} \quad \forall i \in V \tag{16}$$

$$b_i, \Delta b_i \in \{0, 1, \dots, L - l_i\} \quad \forall i \in V \tag{17}$$

$$\Delta ETA_i, \Delta EFT_i \geqslant 0 \quad \forall i \in V \tag{18}$$

$$r_{itq}, r_{it}, u_i, y_{ij}, z_{ij} \in \{0, 1\} \quad \forall i, j \in V, \forall t \in T, \forall q \in R_i \quad i \neq j \tag{19}$$

The objective (1) is the minimization of the overall cost. It includes speeding costs (proportional to the number of time periods for which a vessel is earlier than ETA), tardiness cost (proportional to the number of time periods a vessel departs later than EFT), a one-time penalty cost for finishing later than LFT, and costs related to the QC assignments. The QC assignment cost is a function of the number of QCs and the cost rate per QC-hour. Constraint (2) guarantees that the number of efficient QC hours assigned to a vessel meets the required QC capacity when the deviation factor is taken into account. Constraint (3) ensures that the number of QCs used in each period cannot exceed the available number of QCs. Constraint (4) guarantees that when a vessel is at berth, it will be served by a number of QCs. This constraint ensures that there are no idle periods when the vessel is at port. Constraint (5) represents that the port-stay of a vessel (the right-hand side) is equal to the number of periods the vessel is at berth. Constraint (6) and (7) set the berthing start and end time. The berthing end time is limited by the planning horizon. The deviation from the expected berthing position is calculated by Constraints (8) and (9). Constraints (10) and (12) determine the speed-up time needed to reach s_i, the tardiness of the operations, and whether a penalty must be paid due to a vessel delay. Constraint (13) and (14) help to link the variables y_{ij} and z_{ij}. Constraint (13) ensures that if y_{ij} has a value of one, the berthing position (which corresponds to the fore of the vessel) of vessel $j \in V$ is larger than aft of vessel $i \in V$. Constraint (14) links the vessels if they follow a predecessors relationship in the time frame. If z_{ij} has a value of one, the berthing start time for vessel $j \in V$ is later than the berthing end time for vessel $i \in V$. Constraint (15) ensures either one vessel should be positioned before or after the other in the berth, if not, the berthing intervals should not overlap. Domains of berthing start and end time variables are imposed in (16). Finally, Constraints (18) and (19) set the integer and binary properties of the respective decision variables. A constraint that assigns at most one QC-assignment plan for each time unit and each vessel is added ($\sum_{q \in R_i} r_{itq} \leq 1 \quad \forall i \in V, \forall t \in T$). This constraint is already dominated by Constraint (4). Such a constraint will not improve the LP relaxation, but it can help the Integer Programming (IP) solver to generate useful cuts. The validity of this constraint is evident, since it makes sure that there is at most one kind of QC assignment plan for a certain vessel (i.e. one of $r_i^{min}, r_i^{min}+1, \ldots, r_i^{max}-1, r_i^{max}$) for each period when the vessel is at the berth.

4.1. Valid inequalities and variable fixing methods for the BACAP

To make the mathematical model more efficient, we propose a novel set of valid inequalities, Lower Bounds (LBs) and variable fixing methods. In the remainder of this paper, we refer to this enhanced model as BACAP+.

Before describing the inequalities in details, let us formalize the minimum and maximum processing time of a vessel. The minimum processing time can be obtained when there is no deviation from the desired berthing position, and the maximum number of QCs is assigned to the vessel for every time unit during its berthing interval. In (20), δ^i_{min} is the minimum processing time required to fulfill the QC capacity demand of vessel i. We improve the definition of the minimum processing time, which is Eq. (1) of Meisel and Bierwirth (2009), by imposing the berthing deviation to be zero. Our δ^i_{min} is a more valid lower bound since Meisel and Bierwirth (2009) made a priori value assumption on the berthing deviation.

To calculate the maximum processing time (δ^i_{max}), we need to assume that a vessel i is berthed with the maximum possible deviation from its desired position ($max(L-b_i^0, b_i^0)$), and that the minimum number of QC is assigned throughout its berthing interval. The maximum possible deviation (Δb_i) for vessel i is the maximum of b_i^0 and $L - b_i^0$. In (21), the maximum processing time, δ^i_{max}, is calculated dividing the maximum required QC hours ($m_i(1+\beta max(L-b_i^0, b_i^0))$) by $(r_i^{min})^\alpha$. Since the problem does not allow the preemption of the operations, the use of r_i^{min} in the denominator is required.

$$\delta^i_{min} = \left\lceil \frac{m_i}{(r_i^{max})^\alpha} \right\rceil \qquad \forall i \in V \tag{20}$$

$$\delta^i_{max} = \left\lceil \frac{m_i(1+\beta max(L-b_i^0, b_i^0))}{(r_i^{min})^\alpha} \right\rceil \qquad \forall i \in V \tag{21}$$

We first improve the $Big-M$ in constraint (7). We replace the parameter H with $H-\delta^i_{min}$, and this tightens the best upper bound on the ending time of operations for vessel i. Since this change is trivial, we assume that it is a part of the original BACAP formulation. We now detail the enhancements.

4.1.1. Bounds on s_i

We formulate a class of valid inequalities that aim at improving the lower bound of the s_i (berthing start time) variables. If vessel i is berthed earlier than vessel j, the berthing start time of vessel j (s_j) should be equal-to or greater-than the minimum ending time of vessel i (e_i). Otherwise, the EST still limits the starting time of vessel j.

$$(EST_i + \delta^i_{min})z_{ij} + EST_j(1-z_{ij}) \leqslant s_j \qquad \forall i,j \in V, \quad i \neq j \tag{22}$$

4.1.2. Bounds on processing time

The bounds on the processing time can also be tightened by Constraint (23) using the pre-calculated δ^i_{min} and δ^i_{max} values. Even though these bounds will not serve as valid inequalities, they will help the IP solver to cut-off integer solutions which cannot be optimal.

$$\delta^i_{min} \leqslant \sum_{t \in T} r_{it} \leqslant \delta^i_{max} \qquad \forall i \in V \tag{23}$$

4.1.3. Decomposition of s_i and e_i

In this family of valid inequalities, we improve the link between the QC assignment and the berthing start/end time variables. We observe that reformulating berthing start and end times variables for each time period can help to obtain a better link with the QC assignment variables. We propose a new intermediate binary decision variable denoted $\gamma_{it} = \{0,1\}$. If operations on vessel i starts at period t, the variable is one, and it is zero otherwise. We propose seven inequalities ((24)–(30)) to improve the formulation. First, we guarantee that there can only be one berthing start period for each vessel (Constraint (24)), and this period cannot be before EST (Constraint (25)). Then, the link between γ_{it} and s_i is formulated by Constraints (26) and (27). Constraint (28) guarantees that if a period is a starting period, then the previous period cannot hold any QC assignment. We can also formulate an alternative version of Constraint (28). If a given period is a berthing start period, all previous periods should hold no QC assignment. If that period is not a starting period, no information can be derived for previous periods. Constraint (29) represents it. Finally, Constraint (30) guarantees that if there is a period in which there is a QC assignment for a given vessel ($r_{it} = 1$), then either it is the starting period ($\gamma_{it} = 1$) or the QC operations have already been continuing from the previous period ($r_{it-1} = 1$).

$$\sum_{t \in T} \gamma_{it} = 1 \qquad \forall i \in V \tag{24}$$

$$\gamma_{it} = 0 \qquad \forall i \in V, \forall t \in \{T | t < EST_i\} \tag{25}$$

$$\sum_{t \in T} t\gamma_{it} = s_i \qquad \forall i \in V \tag{26}$$

$$r_{i0} = \gamma_{i0} \qquad \forall i \in V \tag{27}$$

$$r_{it-1} + \gamma_{it} \leq 1 \qquad \forall i \in V, \forall t \in \{T | t > EST_i\} \tag{28}$$

$$\sum_{t' \in \{EST_i, \dots, t-1\}} r_{it'} + (\gamma_{it} - 1)H \leq 0 \qquad \forall i \in V, \forall t \in \{T | t > EST_i\} \tag{29}$$

$$r_{it} \leqslant r_{it-1} + \gamma_{it} \qquad \forall i \in V, \forall t \in \{T | t \geqslant EST_i\} \tag{30}$$

We also use a symmetrical formulation for the end of operations, where $\vartheta_{it} = \{0, 1\}$ is an auxiliary binary variable which is one if operations of vessel i end at period t, and zero otherwise. The difference from the previous inequalities is in the linking constraints. Constraint (37) guarantees that if a QC assignment exists in one period for a vessel ($r_{it-1} = 1$), then either the next period is the ending period ($End_{it} = 1$), or there are still QC operations going on the vessel for the next period ($r_{it} = 1$). Constraints (31)–(36) can be interpreted in the same way as Constraints (24)–(29).

$$\sum_{t \in T \cup \{H\}} t\vartheta_{it} = e_i \qquad \forall i \in V \tag{31}$$

$$\sum_{t \in T \cup \{H\}} \vartheta_{it} = 1 \qquad \forall i \in V \tag{32}$$

$$\vartheta_{it} = 0 \qquad \forall i \in V, \forall t \in \{T \cup \{H\} | t < EST_i + \delta^i_{min}\} \tag{33}$$

$$r_{iH-1} = \vartheta_{iH} \qquad \forall i \in V \tag{34}$$

$$r_{it} + \vartheta_{it} \leq 1 \qquad \forall i \in V, \forall t \in \{T \cup \{H\} | t > 0\} \tag{35}$$

$$\sum_{t' \in \{t, \dots, H\}} r_{it'} + (\vartheta_{it} - 1)H \leq 0 \qquad \forall i \in V, \forall t \in \{T \cup \{H\} | t > EST_i\} \tag{36}$$

$$r_{it-1} \leqslant r_{it} + \vartheta_{it} \qquad \forall i \in V, \forall t \in \{T \cup \{H\} | t > 0\} \tag{37}$$

A similar modeling approach has been suggested for the integrated tactical berth and yard template design problem (Zhen et al., 2011).

4.1.4. Set-partitioning inequalities

Iris et al. (2015) formulate a generalized set partitioning problem (GSPP) formulation for the same BACAP. The authors also propose two preprocessors (or column reduction methods) to reduce the number of columns in short computational times. Each column in the formulation by Iris et al. (2015) corresponds to a feasible solution (assignment) of a vessel to a berthing position with a berthing start and end time. After the column reduction is performed, it is proved that many columns (solutions) for each vessel cannot be part of the optimal solution, and thus they are removed. In this paper, we go through all remaining columns for a single vessel (i) and record earliest and latest possible berthing start ($MinS_i$, $MaxS_i$) and end time ($MinE_i$, $MaxE_i$). Similarly, we record minimum and maximum possible berthing positions ($MinB_i$, $MaxB_i$). With these parameters, we bound the decision variables in Meisel and Bierwirth (2009). Technically speaking, these constraints are not valid inequalities as they can cut away feasible integer solutions. This is because the column reduction methods in Iris et al. (2015) were based on objective value considerations. However, the constraints do not cut away the optimal solution and therefore they do not alter the exactness of the model. The remaining columns provide us better bounds on s_i, e_i and it allows us to eliminate many of the r_{it} and r_{itq} variables. Then we formulate following inequalities:

$$MinS_i \leqslant s_i \leqslant MaxS_i \qquad \forall i \in V \tag{38}$$

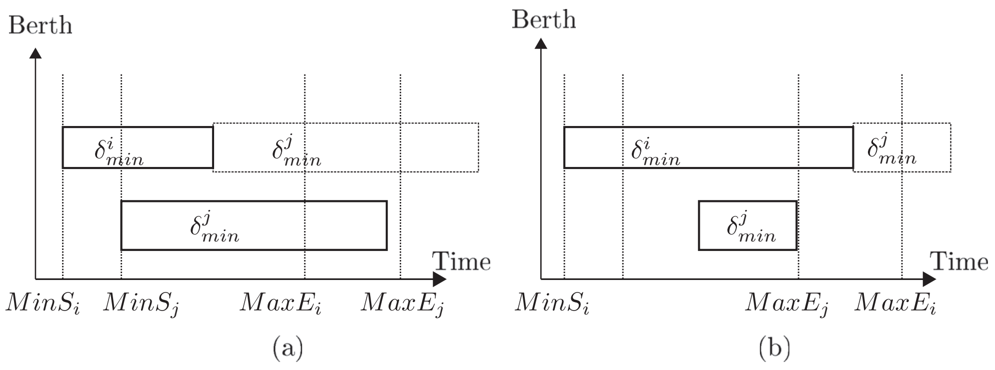

Fig. 2. Different scheduling alternatives for Constraint (44).

$$MinE_i \leqslant e_i \leqslant MaxE_i \qquad \forall i \in V \tag{39}$$

$$MinB_i \leqslant b_i \leqslant MaxB_i \qquad \forall i \in V \tag{40}$$

$$r_{it} = 0 \qquad \forall i \in V, \forall t \in \{T | t > MaxE_i \vee t < MinS_i\} \tag{41}$$

Constraints (38), (39), (40) bound the berthing start time, end time and berthing position of vessel i, respectively. Constraint (41) can eliminate QC assignment variables when period t is greater than maximum feasible berthing end period or smaller than minimum feasible berthing start period for vessel i. [1]

4.1.5. Fixing z_{ij} and y_{ij} variables

By considering minimum and maximum processing times, and the bounds obtained in Section 4.1.4, we can fix and tighten some of the z_{ij}, y_{ij} variables.

$$z_{ij} = 1 \qquad \forall i,j \in \{V | i \neq j \wedge MinS_j \geqslant MaxE_i\} \tag{42}$$

$$y_{ij} = 1 \qquad \forall i,j \in \{V | i \neq j \wedge MaxB_i + l_i \leqslant MinB_j\} \tag{43}$$

Constraint (42) guarantees that if the minimum berthing start time of vessel i is greater-than or equal-to the maximum berthing end time of vessel j, then vessel i is always berthed earlier than vessel j (z_{ij}=1). Constraint (43) works in the same way for the y_{ij} variables.

We also impose that vessel j cannot be berthed later than vessel i if and only if the earliest known berthing start time of vessel i ($MinS_i$), plus the minimum processing times of vessel i and j, is larger than the known maximum berthing end time of a vessel j. This means that these two vessels should either share the quay at different positions or vessel j should end its operations earlier than the berthing start time of vessel i. Fig. 2 illustrates two alternatives of how two vessels, which hold the aforementioned condition, can be feasibly berthed at the same time. This relation is formulated with Constraint (44).

$$z_{ij} = 0 \quad \forall i,j \in \{V | i \neq j \wedge MinS_i + \delta^i_{min} + \delta^j_{min} \geqslant MaxE_j\} \tag{44}$$

Constraints similar to (44) can be formulated for the y_{ij} variables, but the bounds obtained for minimum and maximum possible berthing position are not strong enough to justify their use.

4.1.6. Bound on Δb_i

The last two inequalities focus on the objective function. We should note that these constraints are not valid inequalities, but adding them to the model does not invalidate its exactness. Let us assume that an upper bound of $\bar{z}$ is known for this problem. We can calculate the minimum number of QC hours needed to serve all vessels, $\theta = \sum_{i \in V} r_i^{min} \left\lceil \frac{m_i}{(r_i^{min})^{\alpha}} \right\rceil$. We can extract a LB, denoted by $\underline{z}^1$, on the overall cost as shown in Iris et al. (2015) (See Appendix B for the algorithm to obtain the LB $\underline{z}^1$). Zero deviation is assumed for all vessels in order to calculate the $\underline{z}^1$ (i.e. $\Delta b_i = 0 \quad \forall i \in V$ in $\underline{z}^1$).

We adjust $\underline{z}^1$ by adding the approximated QC operations cost with respect to the deviation from the desired position ($\sum_{i \in V}(c_4(1 + \beta \Delta b_i)m_i)$). In order not to include QC costs twice, we subtract the minimum cost of the QC operations obtained by $c_4\theta$. It follows that the resulting approximation of the objective function must be less-than or equal-to the upper bound value, and this forms Constraint (45).

[1] $\vee$ represents logical OR operator, while $\wedge$ represents logical AND operator.

$$\underline{z}^1 + c_4\left(\sum_{i\in V}(1+\beta\Delta b_i)m_i - \theta\right) \leqslant \bar{z} \tag{45}$$

The term $(1+\beta\Delta b_i)m_i$ is the minimum number of QC hours that we need to spend on vessel i given a deviation of Δb_i from the desired position. However, if we are using more than one QC per hour the actual number of QC hours needed increases because of the interference exponent α. The actual number of QCs used per hour will be determined by the model, but it can never be less than r_i^{min}. Assuming that we use the minimum number of QCs, the number of hours (x) in which we need to operate r_i^{min} QCs to handle a vessel can be found with the following equation.

$$x\left(r_i^{min}\right)^{\alpha} = (1+\beta\Delta b_i)m_i \Rightarrow x = (1+\beta\Delta b_i)m_i/\left(r_i^{min}\right)^{\alpha}$$

We can then conclude that the minimum number of QC hours needed to process vessel i, with a given deviation Δb_i, is;

$$((1+\beta\Delta b_i)m_i/\left(r_i^{min}\right)^{\alpha})r_{min}.$$

This means that we can now impose inequality (46), which is a strengthening of (45) if $\alpha < 1$ and $r_i^{min} > 1$ for all $i \in V$. The only decision variable in Constraint (46) is Δb_i and this constraint will be used in BACAP+.

$$\underline{z}^1 + c_4\left(\sum_{i\in V}\left((1+\beta\Delta b_i)m_i\frac{r_i^{min}}{\left(r_i^{min}\right)^{\alpha}}\right) - \theta\right) \leqslant \bar{z} \tag{46}$$

4.1.7. Bound on time-dependent objective part

As we know that the minimum cost of QC operations is $c_4\theta$, we can set a bound on the objective function for the remaining cost components. The sum of speed-up, lateness and penalty costs should be less than upper bound on overall objective subtracted by LB from QC assignment costs. This is achieved with Constraint (47).

$$\sum_{i\in V}(c_i^1\Delta ETA_i + c_i^2\Delta EFT_i + c_i^3 u_i) + c_4\theta \leqslant \bar{z} \tag{47}$$

4.2. The use of enhancements for relevant studies

The enhancements presented in Section 4.1 can be used not only for the model in Meisel and Bierwirth (2009) but also for a range of other formulations and solution methods. Valid inequalities (22) and (23) consider the minimum and maximum processing time, so these inequalities can be directly used in e.g. the formulations of Hsu (2016), He (2016) (BACAP), Li et al. (2015), Turkogullari et al. (2016) (BACASP), etc. Valid inequalities (24)–(37) can be incorporated into all models that formulate the problem (BAP, BACA(S) P) with the use of integer berthing start and end variables (see Guan and Cheung, 2004; Shang et al., 2016, etc.). This modeling approach is the most dominating method in the state-of-the-art, hence many models can utilize these inequalities. Enhancements (38)–(44) require some parameters that are obtained with the GSPP formulation of the BACAP, and enhancements (46) and (47) require upper bounds and objective functions similar to Meisel and Bierwirth (2009). These enhancements can be used in Meisel and Bierwirth (2013), Li et al. (2015), etc.

5. Adaptive Large Neighborhood Search (ALNS) heuristic for BACAP

ALNS (Ropke and Pisinger, 2006) extends the Large Neighborhood Search (LNS) of Shaw (1998). ALNS is a search algorithm based on a destruction/construction principle. Once an initial solution is found, part of the candidate solution is destroyed by an operator, while keeping the remaining part fixed. A new solution is then found by repairing the destroyed part with a repair operator. These two steps are iterated until some termination criterion. The main difference between ALNS and LNS is that ALNS has multiple destroy/repair operators and the selection of operators is dynamically managed as the search progresses.

To efficiently search the solution space, we generate all candidate assignments for each vessel a priori and apply our ALNS on top of all candidate assignments of all vessels. Each assignment holds a feasible solution of a single vessel. For each vessel $i \in V$, each assignment j holds a cost c_j, a berthing position, berthing start and end time, and a QC plan. The QC plan includes information on how many time units each number of QCs will be used in that assignment. Iris et al. (2015) have proved that there are always two QC numbers which can form a QC plan with minimum cost, these two values are stored in the assignment. The assignment does not give any information about the specific time in which these number of QCs will be used. For example, a QC plan might hold that the assignment will use 3 QCs for 4 time units and 4 QCs for 5 time units. The cost of an assignment includes the time-dependent cost components (speed-up, lateness, penalty costs), and the cost of the QC plan. Since we generate all related information about the berthing start/end times and the berthing position a priori, the time-dependent cost component of an assignment can be calculated easily. The cost of a QC plan can be calculated with the information which is held in the assignment (recalling above example; $(3\cdot4 + 4\cdot5)\cdot c^4$).

A candidate solution for the BACAP (X) is a set of assignments that satisfies all the constraints presented in Section (4). A candidate solution also contains a detailed QC assignment plan which is the number of QCs that will work on each vessel in each period. An optimal solution is a candidate solution with minimum $Z(X)$ where $Z(X)$ accounts for objective function value of a candidate solution X. The set of assignments of vessel i will be denoted by Ω_i, while the set of all assignments is Ω.

The outline of the BACAP ALNS is presented in Algorithm 1. The search requires an initial solution which could be generated by a heuristic or a mathematical model. In this paper, we obtain an initial solution with a construction heuristic. The ALNS attempts to improve this initial solution by removing a subset of assignments from the solution (Note that there is only one assignment of each vessel in a solution) and by inserting new assignments of removed vessels back into the solution sequentially. Since the problem is a minimization problem, if the new solution has an objective value lower than the current best solution, it is labeled as the best solution (line 8). Whenever all vessels which were removed, are reinserted in the current solution, there is an acceptance criterion (line 10) which decides whether the new assignment will be accepted. This procedure continues until a certain termination criterion is met. The weight (i.e. selection probabilities consequently) of each operator is updated periodically (line 12), to promote well-performing ones.

The ALNS has been applied successfully to many optimization problems. In the area of container terminal optimization, Cordeau et al. (2011) have proposed an ALNS for yard assignment problem in a car transhipment terminal. The problem studied in Cordeau et al. (2011) has properties that are related to the BAP; hence our approach uses similar operators (see also Hansen et al., 2008). The authors point out the efficiency of the algorithm on real-life instances. Mauri et al. (2016) have proposed an ALNS for traditional berth allocation problems. The authors have showed the effectiveness of the algorithm on a set of instances. Since our problem covers berth allocation and QC assignment, the operators differ in many ways.

Algorithm 1. BACAP ALNS

```
   Input : An initial solution $X$, initial operator weights
1  $X_{best} \leftarrow X$
2  while termination criteria is not met do
3      $\bar{X} \leftarrow X$
4      Determine the number of vessels ($\phi$) to remove/insert
5      Select and apply one operator to remove assignments of $\phi$ vessels from $\bar{X}$
6      Select and apply one operator to reinsert assignments of $\phi$ removed vessels back into $\bar{X}$
7      if $Z(\bar{X}) < Z(X_{best})$ then
8          $X \leftarrow \bar{X}$, $X_{best} \leftarrow \bar{X}$
9      else
10         if $\bar{X}$ satisfies the acceptance criterion then
11             $X \leftarrow \bar{X}$
12     adjust operator weights
13 return $X_{best}$
```

The main components of BACAP ALNS are operators, adaptive search engine, adaptive weight adjustments, acceptance and termination criterion.

- Operators: To guarantee a diversified search, we propose four destroy and two insertion operators. Details of the operators will follow in this section.
- Adaptive Weight Adjustment: Each operator i is assigned a weight (w_i) and a score (π_i). After each δ iterations, all weights are recalculated. The weight of each operator is updated by considering the score of the operator in the last δ iterations and the current weight value. Initially, all weights are equal to one. While, the scores of all operators are updated after each iteration, and every δ iterations they are all set to zero again. The score of each operator is updated as follows:

$$\pi_i = \begin{cases} \pi_i + \sigma_1 & \textit{if} \quad \text{operator } i \text{ helps to obtain a new best solution of all iterations} \\ \pi_i + \sigma_2 & \textit{if} \quad \text{operator } i \text{ results in a solution which is improving the current objective function} \\ \pi_i + \sigma_3 & \textit{if} \quad \text{operator } i \text{ does not yield a better objective value, the acceptance criterion accepts it} \\ \pi_i & \textit{if} \quad \text{the acceptance criterion rejects the solution} \end{cases}$$

The better an operator performs the higher its score. At end of each iteration, we update the scores of both destroy and repair operators. After the scores have been summed up for δ iterations, the weights can be updated. The weight of operator i is updated in following way:

$$w_i = \begin{cases} w_i & \textit{if} \quad \Psi_i = 0 \\ (1-\eta)w_i + \frac{\eta\pi_i}{\Psi_i} & \textit{if} \quad \Psi_i \neq 0 \end{cases}$$

Let us assume that w_i is the weight of operator i, and Ψ_i is the number of times that the operator was used for the last δ iterations. If the operator is not used, the weight is kept the same, otherwise it is updated. Finally, $\eta \in [0, 1]$ is the reaction factor which reflects the balance between previous weight and its updated value (Ropke and Pisinger, 2006). A low reaction factor makes the weights evolve in a slow and steady fashion.

- Adaptive Search Engine: The selection of which operator to apply is managed by a roulette-wheel technique. The weight of each operator is used to obtain the selection probability of that operator. If there are m operators, operator i would have a selection probability of $w_i / \sum_{j=1}^{m} w_j$. These probability values are used to generate a cumulative probability distribution (CDF) with all operators. A uniform random number from the range [0,1) is drawn, and the inverse of the CDF for that number indicates which operator should be selected.
- Accepting Criteria: Simulated Annealing (SA) has been the most popular acceptance criterion technique for ALNS (see e.g. Ropke and Pisinger, 2006; Muller et al., 2012; Coelho et al., 2012) and it is also used in this study. We accept the new solution s' over the current solution s if s' is better than s. Otherwise, it is accepted with probability $e^{-(f(s')-f(s))/T}$. The term $f(s)$ represents the objective function value (defined by (1)) for solution s. Here, T is the temperature, which is updated by multiplying a cooling factor μ $(0 < \mu < 1)$ at each iteration. Starting from the initial temperature, T_{start}, the temperature is reduced. Instead of specifying a cooling rate parameter, we calculate T_{start}, T_{end} by using the initial solution obtained by the algorithm. We assume that the start temperature is $\varphi\%$ of the initial solution while the ending temperature is $\xi\%$ of initial solution. Then, the cooling factor becomes a parameter which guarantees the convergence from starting temperature to ending temperature after ε iterations. To further diversify the search, temperature T is reheated back to start temperature (T_{start}) every ε iterations. This means that depending on the maximum number of iterations (c_{max}), the algorithm is reheated $\lfloor \frac{c_{max}}{\varepsilon} \rfloor$ times.
- Termination Criterion: The algorithm terminates once a number of iteration (c_{max}) is reached. The selection of each parameter will be discussed in Section 6.1.

In the next subsection, we present the construction heuristic which generates the initial solution. Afterwards, we introduce the operators used in the ALNS heuristic.

5.1. Construction heuristic

Initially, we generate all assignments (Ω), then we select an assignment for each vessel. The initial solution is generated with a greedy approach. First, following a random vessel order, all assignments of each vessel are sorted by their cost in ascending order. Following this order, the vessels' assignments are placed into the solution one by one in a feasible way. If a vessel assignment cannot be inserted, the next assignment for that vessel is attempted. Details of this method will be presented in Section 5.3.

5.2. Destroy operators

Given the chosen destroy operator and a candidate solution X, the operator attempts to add the selected assignment to a destroy list, and it continues until ϕ assignments are added to this list. Once the destroy list, which is a list of assignments to be taken out of candidate solution, is compiled, the algorithm removes these assignments from the candidate solution. This results in a partial solution X_p in which ϕ vessels are missing. This X_p will be the input for the repair operator.

Some destroy operators select an assignment with a probability p (p is a parameter) to introduce some randomness in the selection. This is performed by drawing a random number q in $(0, 1]$ and comparing it with p. If $q \leqslant p$, the current assignment is added to the destroy list; otherwise the next selected assignment is considered and the value of p is increased if necessary.

The number of assignments to be removed (ϕ) is determined by generating a random integer between $\phi_u/2$ and ϕ_u. The value ϕ must satisfy $2 \leqslant \phi \leqslant \phi_u$, where ϕ_u is the upper bound for the number of assignments that is intended to be destroyed. We determine ϕ_u by dividing the size of the instance (e.g. 20, 30 or 40) by a parameter ϕ_r.

If we choose to remove assignments that are not sharing same resources (like the berth or QCs), we might not gain anything when reinserting them. Therefore we now propose a set of removal operators based on different relatedness measures.

5.2.1. Shaw removal

This removal operator was proposed by Shaw (1998). The operator returns a destroy list (D) and a partial solution (X_p). The general idea is to remove assignments that are related, as we expect it to be reasonably easy to shuffle related assignments around and thereby create new, perhaps better, solutions. The BACAP requires a special definition of relatedness. We define the relatedness measure $M(i,j)$ for assignments of i and j in (48).

$$M(i,j) = a|b_i - b_j| + b|s_i - s_j| + c|e_i - e_j| \tag{48}$$

$M(i,j)$ consists of three components: berthing position, berthing start time and end time. The lower value of $M(i,j)$ points a higher relatedness. The steps of the removal can be seen in Algorithm 2. The procedure initially chooses a random assignment to remove (line 1). Then, the relatedness vector is generated for all other assignments in the candidate solution with respect to selected assignment and these assignments are sorted by relatedness in descending order (line 2).

After we generate the sorted list, we use the determinism parameter $p \in (0,1]$ (line 7) to decide whether to put an assignment into the destroy list. In this operator, the parameter p is randomly generated between 0.6 and 1.0, and this gives a relatively high probability of removal for the selected assignment. The parameter p is updated to 1.0 if the number of remaining assignments in a candidate solution is equal to the number of assignments needed to be into the destroy list (line 11). This increase is to ensure that, in the end, exactly ϕ assignments will be in the destroy list. The weights of each factor (a, b, c) are parameters for the algorithm, and they will be tuned in Section 6.1.1.

All of the assignments in the destroy list are removed from the candidate solution (line 13) and the resource availabilities (berth, time, QCs) are updated.

Algorithm 2. Shaw Removal

```
   Input      : $X$, $p \in (0,1]$, $\phi \in \{1,..,N\}$
1  Select an assignment $i$ in $X$ randomly
2  Sort assignments with relatedness to $i$ and generate sorted list
3  counter=0
4  while $|D| < \phi$ do
5      Select the assignment $j$ in the sorted list, draw a random number $q$ in (0, 1]
6      counter++
7      if $q \le p$ then
8          Add $j$ to the destroy list: $D = D \bigcup \{j\}$
9      else
10         if $(\phi - |D| = N - counter)$ then
11             $p = 1$
12     Go to next assignment in the sorted list
13 Remove all assignment in $D$ from the candidate solution $X$
14 $X_p \leftarrow X$
15 return $X_p, D$
```

5.2.2. Cost and time-relatedness removal

This operator first sorts the assignments of the candidate solution in a decreasing cost order. Then it selects the highest cost assignment i and puts it in the destroy list. Afterwards, it searches for time-related assignments of the selected assignment i. Two assignments are time-related if they occupy different berth positions at the same time. All time-related assignments of i are added to the destroy list. If ϕ assignments are not added to destroy list yet, then the next most costly assignment, along with its time-related assignments which are not in the destroy list, are added to destroy list one by one. This procedure continues until ϕ different assignments are collected in the destroy list. The motivation for using the time-relatedness is to increase the likelihood of generating available QC capacity for high-cost vessels.

5.2.3. Cost and berth-relatedness removal

This removal operator is very similar to the one presented in Section 5.2.2. The only difference is the use of berth-relatedness (assignments that share the same berthing positions at different times) instead of time-relatedness. This operator would bring flexibility to reinsert the vessels closer to their desired positions. Note that vessels which are placed closer to the desired position require less QC capacity demand (i.e. shorter processing time).

5.2.4. Random removal

First, an assignment is randomly selected and put into the destroy list. Then, we randomly select new assignments. When a new assignment is randomly selected (after the first one), it is checked whether it is time-related with any of the assignments in the destroy list. If it is time-related, it is immediately put in the destroy list. Otherwise, like line 7 of Algorithm 2, the operator decides whether to put the assignment in the destroy list. In this operator, the parameter p is randomly generated between 0.6 and 1.0 once and updated as in line 10–11 of Algorithm 2.

5.3. Insertion operators

After the destroy operator is executed, we are left with a partial solution. In this partial solution, there are already $N - \phi$ assignments with fixed berthing positions, start and end times, and QC plans. We differentiate between the insertion operator and the insertion list because the insertion list is an input for the insertion operator. The insertion operator uses the order of vessels in the insertion list to insert them back into the partial solution. In this paper, the insertion list is generated by randomly permuting the corresponding vessels of the assignments in the destroy list.

Because the problem is highly constrained, we may end up in situations where there are some vessels that cannot be inserted into the current partial solution. This may happen in any insertion methods. We see such solutions as feasible but we add a high penalty for each unassigned vessel to the objective function in order to make it very attractive to reject such solutions. Two insertion operators have been suggested.

5.3.1. Basic greedy insertion

In order to run the greedy insertion, the insertion list (I), all assignments of the vessels in insertion list ($\Omega_k, \forall k \in I$), the partial solution (X_p) and a determinism factor (p) are required. Algorithm 3 illustrates the steps for the basic greedy insertion.

Algorithm 3. Basic greedy insertion.

```
   Input    : $I, X_p, \phi = |I|, p \in (0,1], \Omega_k : \forall k \in I$
1  for $k = I_1 \rightarrow I_\phi$ do
2      $i = 0$
3      Draw a random number $q$ in (0,1]
4      if $(q > p)$ then
5          $i \rightarrow i+1$, go to 3
6      else
7          if $(fOverlap(i, X_p)=false \wedge fQCcapacity(i, X_p)=enough)$ then
8              Add $i$ to partial solution $X_p$
9              Make detailed QC_Assignment
10             Remove vessel $k$ from insertion list: $I = I - \{k\}$
11         else
12             $i \rightarrow i+1$, go to 3
13 $X \leftarrow X_p$
14 return $X$
```

For each vessel (k) in the insertion list, the algorithm tries to insert the assignments one by one, starting from the one with the lowest cost. An assignment i is considered for insertion with probablity p (line 3–4). Next, the assignment is checked for feasibility (line 7). The function *fOverlap*(i, X_p) checks if assignment i overlaps, with respect to time and berth space, with any assignments in the partial solution X_p. This check is fairly easy since the partial solution includes all information about the berthing positions and intervals. The function *fQCcapacity*(i, X_p) measures whether the QC capacity is enough for the QC assignment. This requires some computational effort. Iris et al. (2015) have proved that there is always an optimum f_1 numbers of $\hat{q}$ QCs and f_2 numbers of $\hat{q}+1$ QCs which minimizes the cost of QC assignment plan and fulfill the QC requirement (i.e. $\sum_{i \in V}(1 + \beta \Delta b_i)m_k$) with interference of α (see Corollary 1 in Iris et al. (2015)). The values of f_1,f_2 and $\hat{q}$ are calculated by the algorithm in Appendix A. Note that, f_1,f_2 represent for how many periods $\hat{q}, \hat{q}+1$ QCs will be used. Hence, $f_1 + f_2$ should be equal to the processing time of that assignment. The values of $\hat{q}$,f_1,f_2 are stored in the assignment. With the values of f_1,f_2 and $\hat{q}$, we can control whether there is enough QC capacity for each period when the vessel will be at berth.

Once assignment i is deemed feasible, the algorithm makes a detailed QC plan with the available information (line 9) since the number of QCs that will work on each vessel in each period is not kept in the assignment. The detailed QC plan works with the principle of "as much as possible". We make this detailed QC plan period by period. For each period from s_i to $s_i + process_i$, we first try to assign $\hat{q}+1$ QCs in that period. If there are not enough free QCs to assign $\hat{q}+1$ QCs for that period, we assign $\hat{q}$ QCs instead. If we already assigned f_2 numbers of $\hat{q}+1$ QCs for that vessel, we just assign $\hat{q}$ QCs in the remaining periods. This procedure continues until a detailed QC plan for that vessel is finalized. The aim is to assign $\hat{q}+1$ QCs as early as possible when there is free QC capacity. We also update the free QC capacities every time a detailed QC plan is made. This is done to evaluate the next assignments accurately. Then we remove the corresponding vessel from the insertion list. This procedure continues until all vessels in the insertion list have been inserted into the partial solution.

We illustrate how a detailed QC plan is generated with a small example. Assume that the vessel that will be inserted has $\{r_k^{min}, r_k^{max}\} = \{3, 5\}$. In this example, we disregard that the interference, and a bad berthing position, can increase the QC capacity demand, which is 18 QC-hours. We assume that the considered assignment does not overlap with any assignments in the partial solution. Moreover, the assignment has a processing time of 5 periods. For the berthing interval of the assignment, Table 2 shows the number of QCs available.

The considered assignment has a $\hat{q}$ value of 3 ($\lfloor \frac{18}{5} \rfloor$) which will be used for 2 periods, and $\hat{q}+1$ (=4) QCs will be used for 3 periods ($3 \cdot 2 + 4 \cdot 3 = 18$). The available QC numbers are adequate for the use of QC plan which is stored in the assignment. We start the genetation of the detailed QC plan starting with the first period to which we assigns 4 QCs. The next period has

Table 2
Available QC numbers – Example 1.

Periods	1	2	3	4	5
QC availability	5	3	5	5	4

Table 3
Detailed QC plan – Example 1.

Periods	1	2	3	4	5
QC plan	4	3	4	4	3

only 3 available QCs, so 3 QCs are assigned to period 2, and so on. In the last period, since we have already assigned 4 QCs in 3 periods, we assign 3 QCs. The resulting detailed QC plan can be seen in Table 3.

Two versions of this insertion operator are used in the ALNS. One version assumes a deterministic insertion ($p = 1$), the other is the stochastic version where a randomly generated parameter p (between 0.25 and 1.0) is used.

5.3.2. Smarter greedy insertion

The operator presented in the previous section can be improved by considering an alternative QC plan. The basic greedy operator uses the values of $\hat{q}, \hat{q}+1$ which form a minimum cost QC plan for a given assignment i of vessel k ($\hat{q} = \left\lfloor \left(\frac{(1+\beta\Delta b_i)m_k}{process_i}\right)^{1/\alpha} \right\rfloor$). High values of $\hat{q}, \hat{q}+1$ can violate the available QC number in a period, and this results in an infeasible QC plan. In such conditions, the basic greedy approach directly skips to the next assignment (which are ordered with increasing cost). The smarter greedy insertion reevaluates the assignment and makes an alternative QC plan for the considered assignment for each time period t.

We illustrate the conditions with another small example. Assume that the vessel that will be inserted has $\{r_k^{min}, r_k^{max}\} = \{2, 6\}$. In this example, we assume that the same conditions holds as the previous example ($\alpha = 1.0, \beta = 0$), and QC capacity demand is 18 QC-hours. For the berthing interval of the assignment, we have the available number of QCs in Table 4.

In the basic greedy operator, the attempted assignment has a $\hat{q} = 3$ ($\lfloor \frac{18}{5} \rfloor$). We cannot make a detailed QC plan based on $\hat{q}, \hat{q}+1$ since the period 3 has only 2 available QCs. The smarter greedy insertion detects this and tries to insert this assignment with an alternative QC plan. Let $QC^t_{available}$ be the QC availability in period t. The smarter greedy insertion first obtains the maximum possible QC number ($QC^t_{possible}$) that can be assigned in time period t by (49).

$$QC^t_{possible} = \min\{QC^t_{available}, r_k^{max}\} \tag{49}$$

Then using $QC^t_{possible}$, the algorithm can determine the maximum number of effective QC hours that can be obtained in the berthing time interval, considering the interference factor. In our example, the calculation is $6^1 + 6^1 + 2^1 + 3^1 + 4^1$ (=21) which is adequate to meet the QC capacity demand (=18). What is more, there are at least r_k^{min} (=2) available QCs in each period during the berthing interval of the assignment. This means that an alternative QC plan is feasible for this assignment.

The alternative QC plan starts by sorting the periods in the berthing interval with increasing $QC^t_{possible}$. The QC plan is done in an iterative fashion, processing one time period at a time, in the sorted order. The algorithm maintains a variable QC_{needed} that indicates the number of QC hours that needs to be provided in the remaining time periods. It starts at the value of QC capacity demand (i.e. $(1+\beta\Delta b_i)m_k$) and the value is decreased after each assignment of QCs. By using QC_{needed} and the remaining number of time periods (p_{remain}) until the berthing end time for that assignment, the algorithm computes a target value for the necessary number of QC for each remaining time period by (50).

$$Target = \left\lceil \left(\frac{QC_{needed}}{p_{remain}}\right)^{\frac{1}{\alpha}} \right\rceil \tag{50}$$

This value seeks to distribute the needed QC hours evenly over the remaining periods. For the vessel k we are trying to insert, it may happen that $Target$ is less than r_k^{min} so we must use $Target^* = \max(Target, r_k^{min})$. Similarly it may not be possible to assign $Target^*$ QCs in the period, because some QCs may be occupied by other vessels. Therefore the actual QC number is obtained by (51) for each t being the time period under consideration.

Table 4
Available QC numbers – Example 2.

Periods	1	2	3	4	5
QC availability	7	6	2	3	4

Table 5
Detailed QC plan – Example 2.

Iteration	Time period (t)	QC_{needed}	P_{remain}	$Target^*$	$QC^t_{possible}$	QC^t_{assign}
1	3	18	5	4	2	2
2	4	16	4	4	3	3
3	5	13	3	5	4	4
4	2	9	2	5	6	5
5	1	4	1	4	6	4

$$QC^t_{assign} = \min\{Target^*, QC^t_{possible}\} \tag{51}$$

After the number of QCs to assign for each period is determined, the needed QC hours (QC_{needed}) is updated by (52).

$$QC_{needed} = QC_{needed} - \left(QC^t_{assign}\right)^{\alpha} \tag{52}$$

Due to the ordering of the time periods, the algorithm will find a feasible insertion if possible. Recalling the above example, we will show how the algorithm makes the updated QC plan. Sorting in the order of increasing $QC^t_{possible}$, it yields the time period ordering 3-4-5-2-1 and Table 5 shows the assignment of the above mentioned variables in each iteration with QC^t_{assign} being the final assignment of QCs (QC plan) in period t.

The updated QC plan (QC^t_{assign}) might be more costly compared to the one stored in the assignment (possibly because it uses more QCs). Therefore, it makes sense to investigate if one of the following assignments is feasible and has a lower cost. As soon as the algorithm reaches an assignment whose stored cost is the same or higher than the produced assignment; the search can stop since the remaining assignments will have the same or a higher cost.

This procedure is applied to all vessels in the insertion list one by one and continues until all vessels are added into the solution. Two versions of this insertion operator are used in the ALNS. One is a deterministic smarter insertion ($p = 1$), the other is a stochastic version where a randomly generated parameter p (between 0.25 and 1.0) is used.

6. Computational results

We compare our results to those that have been obtained in Meisel and Bierwirth (2009), Iris et al. (2015) and Shang et al. (2016). All mathematical models and ALNS are run on a 32 core AMD Opteron at 2.8 GHz and 132 Gb of RAM computer. All running times are reported in seconds.

6.1. Data and experimental settings

There are two benchmarks used in this paper. The improved formulations and ALNS have been tested against state-of-the-art methods by using the benchmark of 30 instances proposed in Meisel and Bierwirth (2009). Furthermore, we also report results on a new and larger benchmark. The size of an instance in the original benchmark (Meisel and Bierwirth, 2009) is defined by the number of vessels (20, 30, 40 vessels). Ten instances of each size are included in the benchmark. For each instance, there are three vessel types (namely feeders, medium, and jumbo) and each of these differs in technical properties, and costs (see Meisel and Bierwirth, 2009 for details). The length of the quay is 1000 meters, which corresponds to 100 berthing positions (with a discretization of 10-meters), and 10 QCs are available. The planning horizon is of 168 working hours (one week), and all planning and assignment operations are based this horizon. The interference coefficient (α) is 0.9, and the increase in the QC-hours needed due to berthing deviation (β) is 0.01.

All models are solved by using CPLEX 12.6.1. We have run the mathematical models under the conditions presented in Meisel and Bierwirth (2009) and Iris et al. (2015). All models are run with a computational time limit of 10 h and allowing CPLEX to use five threads. In order to have a fair comparison, the models presented in Meisel and Bierwirth (2009) and Iris et al. (2015) are re-run. Some of the models in this paper have been warm started with the solution given by one run of the ALNS heuristic. The warmstart solution is fed into CPLEX as the first integer node to initiate the branching procedure. Using a warmstart solution for the BACAP has been first suggested by Iris et al. (2015), where it is shown to achieve better upper bounds for the BACAP.

The ALNS algorithm is implemented in C++, and it uses only one thread. The heuristic is attempted ten times on each instance, and we report best, average, and worst solution. The algorithm is set to terminate after 270000 iterations for each run.

6.1.1. Parameter tuning

First, a set of representative tuning instances is generated. The tuning set consists of 15 instances, and it is generated randomly. The tuning instances are different from the instances used in the following sections. We first introduce the parameters that will be tuned. We start with the removal parameters. The Shaw removal has three parameters (a, b, c). The number of vessels to destroy is managed by ϕ_r, this parameter will also be tuned. The SA is controlled by three parameters, $\varphi, \xi, \varepsilon$. The weight-adjustment is made with four parameters; $\sigma_1, \sigma_2, \sigma_3$ and η. Finally, the update interval of the weights (δ) should also be tuned.

We use a hierarchical procedure for tuning the parameters. First, a phase of parameter settings, which consist of the two reasonable values for each parameter, is decided. We analyze all combinations of these parameters in this phase. Since there are 12 parameters to tune, the first phase runs 2^{12} different parameter settings. This phase results in an initial parameter setting for the ALNS. The first parameter setting results are improved in the second phase. In the second phase, each parameter is tested for four values, while the rest of the parameters are fixed in the best value obtained in the first phase. For the parameters, which we believe are more important ($\phi_r, \varphi, \xi, \varepsilon$), we impose five values to test in the second phase. For each parameter setting, we run the ALNS ten times. This process continues until all parameters have been tuned. The setting that shows the best average objective value is chosen. The complete tuning phase results in a parameter vector of: $(a, b, c, \phi_r, \varphi, \xi, \varepsilon, \sigma_1, \sigma_2, \sigma_3, \eta, \delta)$=(0.01, 2.0, 0.01, 4, 0.05, 0.0005, 30000, 2, 5, 10, 0.8, 1000).

Because parameter tuning aims at obtaining the best parameter setting for the objective function, it may have to compromise on the runtime of the algorithm. We note that some of the parameters directly affect the runtime of the algorithm. For these parameters, we have made an ad-hoc analysis of different values where we analyze the performance of the algorithm on both runtime and objective function. The results show that ϕ_r, which controls how many vessels to remove/insert from/to solution at each iteration, is the most influential parameter on the runtime. Different values of ϕ_r are tested on the tuning instances. The average objective function and average runtime for each setting are presented in Table 6. The results show that ϕ_r can be updated to 6 instead of 4 because the average objective with 4 is slightly better compared to the case where we use 6. However, there is a significant decrease in the computational time.

6.2. Results for improved formulations

We first analyze the performance of the valid inequalities and variable fixing methods on the Linear Programming (LP) relaxation LBs for each instance size. To have a fair comparison, we only add one family of inequality at a time. The results are presented in Table 7 and are split into two main parts. The first part presents the results when CPLEX presolver and cutting planes are included in the LB calculation procedure (1). These options are then disabled, and the LP relaxation LBs are reevaluated in the second part (2). For each column in Table 7, each entry contains two or three numbers. The first number is the average relative gap between the LB and the upper bound obtained by the warmstart solution. The second number (in square parentheses) is the average time in seconds to solve the LP, and the third number (if present) is the time for running the preprocessing algorithms needed for the inequalities. The first column (N) specifies the instance size, the second column (G_R) presents the average gap of the LBs for the original formulation (BACAP without valid inequalities and variable fixing methods).

The columns $V_1, V_2, \ldots, \Theta$ present the results of the formulation with the active inequalities. V_1 refers to the valid inequalities (22), and V_2 refers to (23). V_3 includes the valid inequalities about the start and end decomposition ((24)–(37)). V_4 covers the set partitioning inequalities ((38)–(41)), while V_5 covers the variable fixing inequalities ((42) and (44)). The families of inequalities that focus on the objective function are clustered into a larger group, thus V_6 represents the objective function dependent inequalities ((46) and (47)). The symbol Θ corresponds to all sets of valid inequalities.

Table 6
Parameter ϕ_r vs. solution quality.

ϕ_r	3	4	5	6
Avg. (1)	176.41	175.88	176.35	175.89
Avg. T_k	203.5	150.2	113.0	103.0

Table 7
Performance of LP relaxation LBs for different enhancements.

		CPLEX presolver and cutting planes included (1)						
N	G_R (%)	V_1 (%)	V_2 (%)	V_3 (%)	V_4 (%)	V_5 (%)	V_6 (%)	Θ (%)
20	13.8 [3.4]	12.0 [3.5]	11.6 [3.2]	8.9 [7.5]	12.1 [2.6 + 1.1]	14.8 [3.6 + 1.1]	12.8 [2.9]	9.4 [4.4 + 1.1]
30	26.1 [9.5]	25.6 [11.5]	24.6 [9.6]	20.0 [28.6]	25.5 [12.5 + 2.6]	26.1 [9.4 + 2.6]	25.9 [9.9]	19.9 [15.5 + 2.6]
40	48.9 [52.3]	45.0 [55.1]	46.1 [48.1]	37.9 [79.1]	46.5 [50.1 + 3.1]	47.3 [54.3 + 3.1]	46.1 [58.1]	37.3 [92.1 + 3.1]
		CPLEX presolver and cutting planes disabled (2)						
20	27.2 [2.7]	27.2 [2.9]	27.2 [3.1]	26.8 [3.5]	27.2 [2.6 + 1.1]	27.2 [2.7 + 1.1]	27.2 [2.8]	26.8 [3.4 + 1.1]
30	37.6 [4.3]	37.2 [3.9]	37.5 [4.4]	36.3 [6.4]	37.2 [4.4 + 2.6]	37.4 [4.3 + 2.6]	37.5 [4.3]	35.9 [4.9 + 2.6]
40	54.8 [7.2]	54.7 [7.5]	54.7 [7.6]	53.8 [12.3]	54.6 [7.2 + 3.1]	54.8 [7.1 + 3.1]	54.3 [7.3]	52.8 [12.4 + 3.1]

Gap [CPU time: time to obtain the LB + time to generate parameters (if any)].

The first important observation from Table 7 is that the performances of the inequalities are not that strong when presolver and cutting planes are disabled. However, when these two CPLEX options are activated, CPLEX generates very useful cuts and the average integrality gap is reduced for each set of instances. Another important observation is that including all inequalities (Θ) results in better LBs and this proves the contribution of the inequalities.

Let us now analyze the details of each set of inequalities with respect to the gaps. The performances are evaluated for the case in which CPLEX presolver and cutting planes are kept active. The results show that for small scale instances, the performance of the inequalities is erratic. For medium and large scale instances, the contribution is more clear. In general, by adding all inequalities, the gap is improved by 27% in average (case Θ, last column). Looking at one inequality class at a time, one can observe that valid inequalities V_3 perform better than the rest, and they are followed by V_1 and V_2. The LP relaxation is improved in average by 26% with V_3, while the improvement is 9% with V_2, and 8% with V_1. Individual results for each instance show that each family of inequalities has a significant contribution on different instances. Although the gaps seem to be similar for V_1, V_2, V_5, V_6, each family of inequalities contributes to different instances.

It should be noted that inequalities V_4, V_5 require running the two preprocessing methods (Iris et al., 2015) to obtain the bounds used in these constraints. For this reason, the computational time to generate these bounds (e.g. $MinS_i$, $MaxE_i$, $MaxB_i$, etc.) are added to LP relaxation time. In Iris et al. (2015), the authors also propose two probing methods to improve the LBs further. However, the high computational times needed to obtain these bounds did not justify their use in the BACAP+ formulation.

The LB obtained with one type of inequality is sometimes worse than the one which is obtained with the original BACAP formulation (e.g. Instance 3, V_1). This is explained by the fact that CPLEX uses a heuristic separation procedure for the cuts, and this leads to the generation of a different set of inequalities. With the help of Table 7, we justify using all inequalities in the BACAP+ model.

We now present the complete results of models run with a time limit of 10 h. In these models, the CPLEX presolver and cutting planes options are active. The results of the original model (BACAP), original model with all enhancements (BACAP+) and GSPP model are presented in Table 8. In this table, the column "*N*" shows the number of vessels (instance size), "#" indicates the instance ID. The columns named "*Z*" show the best upper bounds obtained, while "BLB" report the best LBs found within the time limit. The optimality gap ($G = \frac{Z-BLB}{Z}$) is calculated between the upper and lower bounds. The columns "T_c" show the time spent to solve the mathematical model, while "R_{LB}" reports the LBs which are also in Table 7.

In Table 8, column "BACAP model" presents the results obtained from the paper of Meisel and Bierwirth (2009). These results do not include any enhancements or warmstart. Column "BACAP with warmstart" shows the rerun results of the original (Meisel and Bierwirth, 2009) model without any enhancements, and with a warmstart. Column "BACAP+" presents results for the model with all enhancements (BACAP+) and warmstart using the default CPLEX settings, while column "BACAP+ (Cuts&Opt. Emph)" presents the results of the model with all enhancements (BACAP+) and warmstart using the CPLEX option of *emphasize optimality* and *aggressive cut generation*. Finally, column "GSPP" presents the results of reruns of the GSPP model (Iris et al., 2015) with the warmstart. All mathematical models analyzed in this paper (BACAP, BACAP +, GSPP) run with the same warmstart for each instance for fair comparison.

The model of Meisel and Bierwirth (2009), if run without the warmstart procedure, is only able to achieve optimal solutions for four instances, while it cannot generate any integer solutions for almost all medium and large-scale instances.

Results show that the BACAP+ model produces better optimality gaps for all the benchmark instances and thus outperforms the original one (BACAP). For those instances where the BACAP model is also able to find optimal solutions, the BACAP+ model always needs less computational time. The use of the enhancements has reduced the average optimality gap by 76% compared to the original BACAP model. Additionally, almost all upper and lower bounds in the literature have been improved by BACAP+ model with warmstart. The results show that all small and medium scale ($N = 20$, 30 vessels) instances are solved with less than 2% optimality gap by the BACAP+ formulations. For larger instances ($N = 40$ vessels), the average optimality gap is 8.2%. The BACAP+ model also remains competitive with the GSPP model. In average, the BACAP+ model performs better in both small, medium and large-scale instances. However, the number of instances which are solved to optimality is higher for the GSPP model. The GSPP model solves thirteen instances to optimality, while nine instances are solved with the BACAP+ models. In Table 8, we mark the gaps and/or computational times which are better compared to other results.

The comparison of different CPLEX settings is also relevant. The results show that the average gap is reduced activating the "emphasizing optimality" and "aggressive cut generation" options. With these settings, CPLEX obtains the same or better optimality gaps compared to the default settings in 27 out of 30 instances. BACAP+ with the special settings also obtains a better root node LB compared to the default settings in many instances. However, the number of found optimal solutions does not change.

6.3. Results for ALNS

The computational performance of the ALNS is compared to the best known upper bounds and three state-of-art heuristics: Squeaky Wheel Optimization (SWO) and Tabu Search (TS) from Meisel and Bierwirth (2009), and Genetic Algorithm from Shang et al. (2016). There are other heuristics in those paper, but the authors have shown that these selected heuristics (SWO, TS, GA) outperform the others. The ALNS algorithm is run 10 times for each instance. The average, best, and worst

Table 8
Comparison results of Model by Meisel and Bierwirth (2009), GSPP model by Iris et al. (2015), BACAP+.

N	#	BACAP Model			BACAP with warmstart					BACAP+ with warmstart					BACAP+ (Cuts&Opt. Emph.)					GSPP with warmstart				
		Z	BLB	G [%]	Z	BLB	G [%]	T_C	R_{LB}	Z	BLB	G [%]	T_C	R_{LB}	Z	BLB	G [%]	T_C	R_{LB}	Z	BLB	G [%]	T_C	R_{LB}
20	1	84.1	84.0	0.12	84.1	84.1	0.0	453	70.2	84.1	84.1	0.0	**14**	77.8	84.1	84.1	0.0	28	77.2	84.1	84.1	0.0	101	83.3
	2	53.9	53.9	0.0	53.9	53.9	0.0	3	50.3	53.9	53.9	0.0	2	51.0	53.9	53.9	0.0	5	53.6	53.9	53.9	0.0	**1**	53.9‡
	3	77.4	75.2	2.93	76.3	76.0	0.32	36,000	66.2	76.3	76.1	0.25	36,000	68.3	76.3	76.1	0.18	36,000	69.0	76.3	76.3	**0.0**	**245**	75.3
	4	76.2	75.8	0.52	76.2	75.9	0.37	36,000	66.0	76.2	76.2	**0.0**	510	68.2	76.2	76.2	**0.0**	8072	70.3	76.2	76.2	**0.0**	**390**	74.4
	5	56.8	56.8	0.00	56.8	56.8	0.0	12	49.7	56.8	56.8	0.0	**10**	52.6	56.8	56.8	0.0	13	51.2	56.8	56.8	0.0	32	56.8‡
	6	57.6	57.6	0.00	57.6	57.6	0.0	2	57.6	57.6	57.6	0.0	2	56.0	57.6	57.6	0.0	4	56.2	57.6	57.6	0.0	**1**	57.6‡
	7	68.0	67.5	0.74	68.0	68.0	0.0	445	57.8	68.0	68.0	0.0	58	61.6	68.0	68.0	0.0	129	65.1	68.0	68.0	0.0	**12**	67.1
	8	56.1	56.1	0.00	56.1	56.1	0.0	26	51.0	56.1	56.1	0.0	**13**	51.8	56.1	56.1	0.0	22	51.9	56.1	56.1	0.0	21	55.9
	9	75.1	75.0	0.13	75.1	75.1	0.0	324	62.0	75.1	75.1	0.0	**68**	67.0	75.1	75.1	0.0	101	66.2	75.1	75.1	0.0	158	73.2
	10	90.9	88.2	3.06	89.3	88.9	0.36	36,000	65.6	89.3	89.1	**0.18**	36,000	72.8	89.3	89.1	**0.18**	36,000	73.6	89.3	87.8	1.68	36,000	83.0
							0.10					0.04					0.03					0.16		
30	11	X	137.7	–	140.4	134.8	3.93	36,000	95.8	139.8	139.6	0.09	36,000	106.5	139.8	139.6	**0.08**	36,000	108.1	139.8	139.4	0.22	36,000	133.8
	12	81.8	81.4	0.49	81.8	81.8	0.0	478	74.3	81.8	81.8	0.0	**35**	74.3	81.8	81.8	0.0	74	74.3	81.8	81.8	0.0	41	81.8‡
	13	104.9	100.9	3.96	102.4	101.8	0.55	36,000	88.4	102.4	102.1	0.28	36,000	88.6	102.4	102.1	0.29	36,000	89.3	102.4	102.4	**0.0**	**792**	100.8
	14	X	96.8	–	99.1	98.3	0.79	36,000	81.3	99.1	98.7	0.33	36,000	85.2	99.1	98.8	0.24	36,000	85.7	99.1	99.1	**0.0**	**2558**	97.5
	15	X	136.9	–	150.4	135.4	9.94	36,000	91.9	150.4	149.5	**0.55**	36,000	104.6	150.4	148.4	1.27	36,000	106.8	150.4	148.5	1.20	36,000	136.8
	16	X	106.2	–	113.8	110.7	2.64	36,000	86.1	113.8	113.2	0.47	36,000	90.5	113.8	113.4	**0.31**	36,000	91.3	113.9	111.8	1.78	36,000	107.8
	17	X	99.6	–	102.6	102.0	0.58	36,000	74.8	102.6	102.2	0.39	36,000	87.1	102.6	102.3	0.24	36,000	88.7	102.6	102.6	**0.0**	**403**	100.7
	18	X	117.8	–	121.9	118.9	2.44	36,000	93.6	121.9	121.5	**0.32**	36,000	104.8	121.9	121.4	0.42	36,000	104.0	121.9	120.5	1.13	36,000	117.4
	19	X	156.4	–	165.2	143.3	13.2	36,000	108.9	165.2	162.1	1.91	36,000	129.8	165.2	163.3	**1.11**	36,000	127.0	165.2	162.2	1.82	36,000	158.8
	20	X	125.6	–	132.5	121.6	8.21	36,000	97.5	131.5	131.0	0.36	36,000	103.6	131.5	131.1	**0.29**	36,000	104.8	131.6	129.7	1.41	36,000	128.6
								4.22					0.47					0.42					0.75	
40	21	X	165.7	–	201.9	148.4	26.4	36,000	100.0	201.9	178.7	11.48	36,000	137.6	201.9	181.3	**10.15**	36,000	137.9	203.0	179.4	11.60	36,000	178.0
	22	X	159.6	–	170.9	152.2	10.91	36,000	110.9	170.9	163.8	4.12	36,000	127.1	170.9	165.3	3.25	36,000	122.9	171.0	166.8	**2.44**	36,000	163.1
	23	X	185.0	–	239.5	162.2	32.21	36,000	100.3	234.2	199.8	14.68	36,000	122.2	234.2	205.0	**12.47**	36,000	121.1	264.4†	196.7†	25.60†	36,000	196.4
	24	X	224.1	–	284.0	193.9	31.37	36,000	119.3	282.5	239.0	15.39	36,000	152.1	282.3	243.5	**14.04**	36,000	152.0	302.2†	245.4†	18.80†	36,000	239.4
	25	X	133.3	–	148.7	126.8	14.71	36,000	104.1	148.9	141.0	5.29	36,000	110.7	148.7	141.6	4.79	36,000	111.4	148.1	143.4	**3.13**	36,000	139.9
	26	X	201.3	–	238.6	166.0	30.40	36,000	96.4	230.7	208.5	9.61	36,000	135.0	229.3	211.6	**7.71**	36,000	135.1	245.0†	208.8†	14.80†	36,000	208.0
	27	X	172.2	–	183.8	149.2	18.79	36,000	97.8	183.9	173.9	5.41	36,000	135.9	183.9	175.8	**4.40**	36,000	139.8	188.0	174.0	7.41	36,000	173.9
	28	X	211.7	–	260.9	161.8	37.89	36,000	102.7	262.1	204.1	22.11	36,000	156.2	262.1	218.9	**16.47**	36,000	156.1	315.2†	215.3†	31.70†	36,000	215.3
	29	X	180.3	–	207.0	152.6	26.24	36,000	123.5	207.0	193.5	6.48	36,000	126.7	207.1	194.9	**5.87**	36,000	126.7	208.8	185.3	11.24	36,000	185.5
	30	X	170.1	–	190.9	163.6	14.30	36,000	117.3	185.8	174.2	6.21	36,000	131.0	185.1	179.5	**3.01**	36,000	131.7	187.8†	175.0†	6.80†	36,000	173.9
							26.05					10.07					8.21					13.35		

† represents that CPLEX solver has run out of memory while the instance was running using 5 threads. Hence, ‡ are results of Iris et al. (2015), ‡ represents that the instance is solved to optimality in the root node. The bold font is used to indicate the best approach for each instance. If all models solve the instance to optimality, the best solution time is marked. If the instance is not solved to optimality by any models, the best gap is marked.

Table 9
ALNS results.

		ALNS								SWO			TS			GA		
#	Z^*	Average cost	Best cost	Worst cost	Average G (%)	Best G (%)	ALNS-[B] G (%)	T_{k1}	T_p	Cost	T_{k2}	G (%)	Cost	T_{k2}	G (%)	Cost	T_{k2}	G (%)
1	84.1†	**84.8**	84.1	85.1	0.83	0.00	−0.35	13.8	0.6	85.1	11	1.18	85.1	14	1.18	90.5	10	7.07
2	53.9†	**53.9**	53.9	53.9	0.00	0.00	0.00	2.5	0.5	**53.9**	4	0.00	**53.9**	8	0.00	56.8	8.4	5.10
3	76.3†	78.2	77.4	78.4	2.38	1.42	0.97	30.1	0.4	**77.4**	11	1.42	**77.4**	17	1.42	82.7	10.9	7.73
4	76.2†	79.7	77.9	87.7	4.39	2.18	2.26	18.6	0.6	79.7	8	4.39	**77.9**	12	2.18	80.7	10.1	5.57
5	56.8†	57.0	56.8	57.7	0.32	0.00	0.32	8.9	0.4	**56.8**	10	0.00	**56.8**	15	0.00	**56.8**	8.9	0.00
6	57.6†	**57.6**	57.6	57.6	0.00	0.00	0.00	2.0	0.5	**57.6**	4	0.00	**57.6**	7	0.00	**57.6**	9	0.00
7	68†	69.0	68.3	69.2	1.38	0.44	0.07	6.1	0.5	**68.9**	17	1.31	**68.9**	24	1.31	76.8	9.9	11.45
8	56.1†	58.0	56.1	59.9	3.28	0.00	3.28	6.8	0.5	57.0	8	1.57	**56.1**	13	0.00	57.2	9.1	1.92
9	75.1†	**75.3**	75.3	75.4	0.29	0.27	−0.24	7.6	0.5	75.9	18	1.05	75.5	14	0.53	76.4	10.2	1.70
10	89.3	**92.2**	90.1	94.6	3.19	0.89	−0.82	13.9	0.6	94.6	10	5.60	93	10	3.98	96.3	10.3	7.26
			G_{av}		1.60	0.52	0.55	10.9			10.1	1.65		13.4	1.06		9.68	4.78
11	139.8	**146.6**	143.3	147.8	4.61	2.44	−0.85	68.6	0.8	147.8	51	5.41	149.5	61	6.48	168.2	29.3	16.88
12	81.8†	**82.4**	82.4	82.4	0.73	0.73	−0.12	8.1	0.7	83.3	17	1.80	82.5	36	0.85	84.0	29.9	2.61
13	102.4†	105.4	104.7	105.6	2.81	2.20	0.82	14.1	0.7	105.7	53	3.12	**104.5**	41	2.00	105.2	33.1	2.66
14	99.1†	104.1	103.3	105.6	4.76	4.07	1.20	47.5	0.7	105.8	22	6.33	113.2	41	12.45	**102.8**	32.4	3.59
15	150.4	**154.2**	152.4	157.0	2.49	1.31	−2.07	45.8	0.8	159	57	5.41	157.4	79	4.44	159.3	36.6	5.58
16	113.8	**116.5**	116.4	117.6	2.32	2.23	−1.72	46.2	0.9	118.5	40	3.97	119.5	51	4.76	120.0	37.3	5.16
17	102.6†	**104.0**	103.5	104.2	1.31	0.87	−0.23	24.5	0.7	104.5	38	1.82	104.2	41	1.53	107.0	33.8	4.11
18	121.9	**125.1**	124.6	125.6	2.59	2.17	−0.32	34.1	0.8	125.5	20	2.87	131.2	46	7.07	136.8	34.6	10.89
19	165.2	**171.3**	170.1	173.1	3.54	2.88	−1.48	58.8	1.0	173.8	27	4.95	173.8	41	4.95	176.1	43.9	6.18
20	131.5	136.2	135.3	136.3	3.44	2.81	0.73	48.7	0.8	**135.2**	58	2.74	138.3	93	4.91	139.1	37.2	5.46
			G_{av}		2.86	2.17	−0.40	39.9			38.4	3.85		53	4.94		34.8	6.31
21	201.9	**208.6**	204.6	210.3	3.19	1.32	−3.09	133.6	1.2	215	311	6.09	226.7	209	10.93	225.8	100.6	10.58
22	170.9	179.5	178.5	180.3	4.81	4.26	0.41	83.4	0.9	**178.8**	163	4.42	183.4	165	6.81	198.1	75.4	13.73
23	234.2	**251.4**	244.6	263.0	6.84	4.25	−5.13	163.7	1.0	273.9	315	14.49	264.3	373	11.39	286.2	115.6	18.16
24	282.5	**290.2**	285.9	294.9	2.73	1.26	−12.54	143.7	1.1	326.6	325	13.56	342.2	351	17.44	326.8	110	13.55
25	148.1	**154.8**	152.9	155.9	4.32	3.14	−0.01	87.1	0.8	155.1	206	4.51	154.8	140	4.33	162.6	98.1	8.91
26	229.3	**249.1**	245.9	256.7	7.93	6.75	−4.23	183.8	1.0	260.4	130	11.94	259.6	298	11.67	290.0	106.5	20.93
27	183.8	**191.9**	189.1	193.5	4.21	2.80	−4.65	145.3	0.9	200.8	209	8.47	215.8	282	14.82	215.3	106.2	14.63
28	260.9	**272.9**	263.4	286.3	4.39	0.95	−4.88	208.1	1.2	286.2	373	8.84	294.3	109	11.34	351.1	139.6	25.69
29	207	**215.5**	209.9	222.9	3.92	1.38	−1.83	71.3	1.0	219.4	202	5.65	223.4	175	7.34	225.0	96.1	8.00
30	185.1	**202.0**	194.9	207.8	8.35	5.03	−19.28	128.0	1.1	240.9	209	23.16	254.7	395	27.32	238.8	109.3	22.48
			G_{av}		5.07	3.11	−5.52	134.9			244.3	10.11		249.7	12.38		105.7	15.66

$G_{av} = \left[\frac{\sum_{i \in n} G_i}{n}\right]$: Average gap or computational time for each instance size. The bold font is used to mark the best heuristic value for each instance. The bold font is based on average ALNS results for 10 runs instead of the best ALNS results, this is to have a fair comparison.

† The Z^* is proved to be the optimal solution.

objective values of ALNS for 10 runs, all relevant gaps, average run times, and the time for generating all assignments are reported for each instance.

In Table 9, column "Z^*" shows the best upper bound known in the literature. Then next seven columns report the results for the ALNS. The table also includes the results of SWO and TS of Meisel and Bierwirth (2009), and GA of Shang et al. (2016).

In Table 9, the average gap is calculated using the average ALNS cost and Z^* (Average G=$\frac{\text{Average Cost}-Z^*}{\text{Average Cost}}$, column 6), the best gap is calculated using the best ALNS cost and Z^* (Best G=$\frac{\text{Best Cost}-Z^*}{\text{Best Cost}}$, column 7). The gap between the best result among SWO, TS, GS and the average ALNS result is ALNS-[B] Gap ($\frac{\text{Average Cost}-\text{best of SWO/TS/GA cost}}{\text{Average Cost}}$, column 8). A negative value of this gap shows the ALNS performs better compared to Meisel and Bierwirth (2009) and Shang et al. (2016) best results. While, T_{k1} is the average computational time of the ALNS and T_{k2} is the computational time for SWO, TS or the GA heuristics. T_p is the time to generate the set of assignments (Ω) which is considered as part of the ALNS. The gaps for SWO, TS and GA are calculated using the respective cost value and Z^* (G=$\frac{\text{Cost}-Z^*}{\text{Cost}}$, columns 13, 16, 19). Note that the gap calculations are based on two decimals places of the ALNS cost values, while we report a single decimal cost values in Table 9.

The results show that the average gaps of the ALNS from the best-known solution for small, medium, and large-scale instances are 1.6%, 2.86%, and 5.07% respectively. These results support that the ALNS heuristic obtains good results in short computational times. If we look at individual instance, there are five instances for which ALNS obtains the optimal solutions. For the majority of the instances, the ALNS is able to find the best known heuristic solutions. The improvement from average solution to best solution is 1.27% in average.

The results also show that average gaps of the ALNS are mostly better than the heuristics presented in Meisel and Bierwirth (2009) and Shang et al. (2016). The average ALNS results are better or equal compared to the best of SWO/TS/GA results in 21 of 30 instances. When the best solution of the ALNS is compared to the best solutions of the SWO/TS/GA, the ALNS is better in 28 of 30 instances. In average, the average ALNS results outperform the state-of-the-art heuristics for medium and large-scale instances. The results also show that the ALNS is stronger as the instance size increases.

When comparing the ALNS with the other heuristics (SWO, TS and GA), we should also emphazise the fact that the results are mostly obtained in comparable computational times (Note that SWO and TS results were obtained using a 2.4 GHz Pentium IV, which is assumed to be slower compared to our computer, while GA results obtained using a 64 core AMD Opteron at 2.1 GHz). We also report the time to generate the set of assignments which is at most 1.2 s. In average, ALNS consistently outperforms SWO and GA on gap, ALNS also outperforms SWO with respect to computational time. The same comment holds for TS except TS obtains a better gap in small-scale instances.

The computational time and the optimality performance tradeoff of the ALNS are strongly affected by the number of iterations and the parameter ϕ_r (which controls how many vessels to be removed/reinserted at every iteration). For this reason, we perform further tests that aim at pointing out the performance of the heuristic for different values of these two parameters. For the remaining parameters, the parameter tuning results are used. All further performance analyses are done using the average results over 10 runs.

Fig. 3 illustrates the average gap (a) and the average computational time (b) for a different number of iterations. The results are clustered for each instance size. The average gap of all instance sizes converges strongly after approximately 450000 iterations. The average computational time linearly increases as the number of iteration increases. The best average gaps obtained are approximately 4.7%, 2.7%, and 1.5% for large, medium and small-scale instances. Fig. 3 supports the selection of 270000 iterations as the termination criterion since the gap improvements are minimal after this value, and the computational times are also reasonable.

Fig. 4 illustrates the average gap (a) and the average computational time (b) for different values of the ϕ_r parameter. This removal parameter adversely affects the number of vessels that will be removed/inserted at every iteration. The higher value

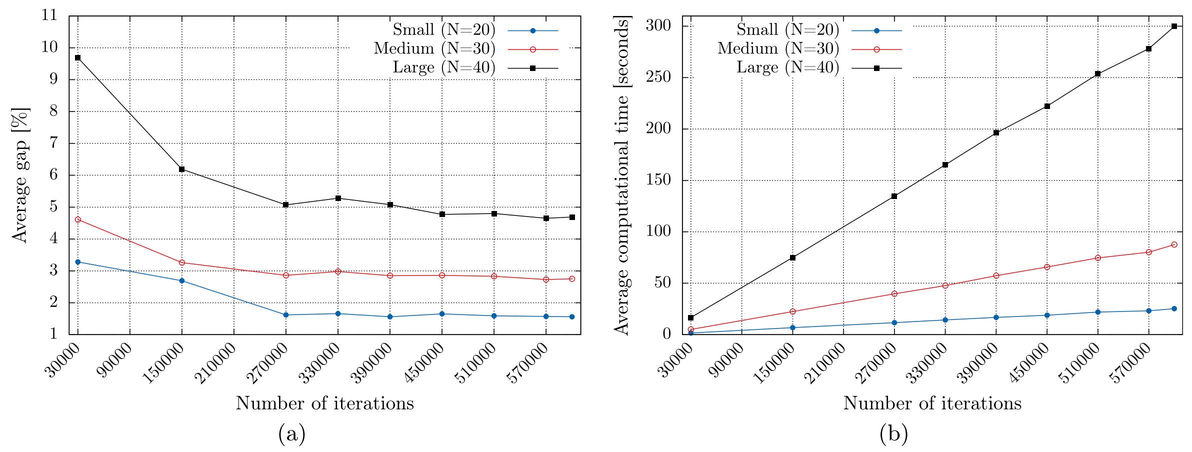

Fig. 3. Analysis of number of iteration.

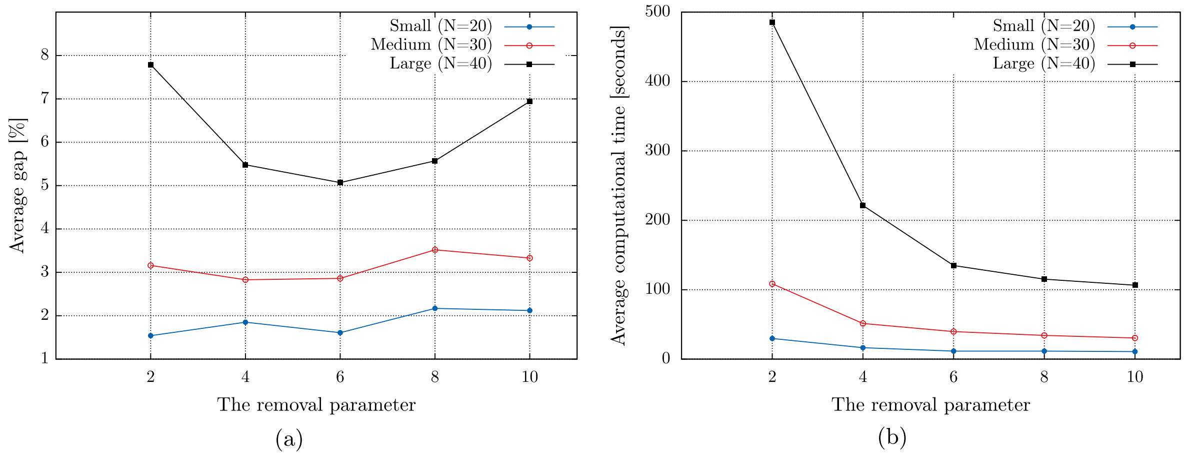

Fig. 4. Analysis of ϕ_r.

Table 10
Impact of insertion operators on the ALNS performance.

Greedy insertion		Smarter greedy insertion		Increase in the average cost
deterministic	stochastic	deterministic	stochastic	(%)
1	0	0	0	2.81
0	0	1	0	1.70
1	1	0	0	2.96
0	0	1	1	1.85

of ϕ_r the less the number of vessels to be removed, and vice versa. Figs. 4a and b are drawn for each instance size. Fig. 4a shows that the least average gap is obtained for ϕ_r values of 6 for large, 4 for medium and 2 for small-scale instances. Fig. 4b points out that the average computational time decreases as the value of ϕ_r increases. With the help of the two sub-figures, we can state that the tradeoff between computational time and optimality performance of ALNS has a break-even point at $\phi_r = 6$.

Finally, we also identify how well each insertion operator performs if they are used as the only insertion operator. We report the average increase in the average solution cost (%) for different combinations of the insertion operators in Table 10. The first observation is that each combination increases the average objective value, which means that the use of all insertion methods together has a contribution to the ALNS algorithm. The results also show that the use of deterministic smarter greedy insertion as the only insertion method, yields the least amount of increase in the average objective value and the smarter greedy insertion consistently outperforms the basic greedy insertion method in both deterministic and deterministic + stochastic versions.

6.4. Computational results for larger instances

Depending on their size, container terminals handle a different number of vessels per week. Because large container terminals are not completely represented in the current state-of-the-art benchmark, we have generated a new benchmark of twenty instances (ten instances with 60 vessels, and the other ten with 80 vessels). We generated the benchmark using parameters which are similar to those described in Meisel and Bierwirth (2009). The parameters are presented in Table 11 along with the estimated/realized berth and QC utilization. In Table 11, the number of vessels (N), QCs (Q) and the quay length (L) are shown in the first three columns. Following that, the estimated and realized berth/QC utilization are shown as Key Performance Indicators (KPIs). The estimated KPIs are calculated using the parameter values in the benchmark, while the realized KPIs are calculated with the results of the representative instances. Estimated berth utilization is found by dividing the estimated berth-hours use by the available berth-hours. As the weighted average vessel length is $l_{av} = 19.85$ and the weighted average processing time (without any berthing deviation) is $p_{av} = 9.2$, the estimated berth-hours use is $N \cdot l_{av} \cdot p_{av}$. Dividing the estimated berth-hours use by the available berth hours ($L \cdot T$), one can obtain the estimated berth utilization. The estimated QC utilization is calculated dividing the estimated QC-hours use by the available QC-hours. The estimated QC-hours use is calculated multiplying the number of vessels (N) with the weighted average QC number per period ($r_{av} = 2.3$) and the weighted average processing time without any deviation ($p_{av} = 9.2$), while the available QC-hours is $Q \cdot T$.

Results in Table 11 show that the new benchmark has a higher berth and QC utilization compared to the benchmark of Meisel and Bierwirth (2009). The new benchmark has a range of 65–80% utilization, which closely reflects reality for some

Table 11
Benchmark characteristics, berth and QC utilization.

N	Q	L	Estimated berth-hours use	Available berth-hours	Estimated berth uti. (%)	Realized berth uti. (%)	Estimated QC-hours use	Available QC-hours	Estimated QC uti. (%)	Realized QC uti. (%)
20	10	100	20·19.85·9.2	100·168	21.7	21.3	20·2.3·9.2	10·168	25.2	31.4
30	10	100	30·19.85·9.2	100·168	32.6	36.4	30·2.3·9.2	10·168	37.8	50.4
40	10	100	40·19.85·9.2	100·168	43.4	44.5	40·2.3·9.2	10·168	50.4	62.3
60	12	100	60·19.85·9.2	100·168	65.2	68.8	60·2.3·9.2	12·168	62.9	79.6
80	16	150	80·19.85·9.2	150·168	57.9	57.8	80·2.3·9.2	16·168	62.9	74.7

Table 12
Computational results for new benchmark.

N	#	BACAP+				ALNS					
		Z	LB	G	T_C	Average cost	Best cost	Worst cost	T_{k1}	T_P	G_{ALNS}
60	31	–	188.3	–	36,000	334.9	321.6	352.8	456.6	1.5	43.8
	32*	–	203.4	–	17,875	365.7	349.2	389.0	347.1	1.4	44.3
	33*	–	167.7	–	32,124	218.9	215.4	223.8	243.6	1.2	23.3
	34	–	217.9	–	36,000	329.5	318.4	336.4	413.2	1.4	33.8
	35	–	222.1	–	36,000	357.1	348.3	375.8	408.2	1.8	37.8
	36	–	214.3	–	36,000	572.3	542.3	608.1	612.4	1.6	62.5
	37	–	213.4	–	36,000	472.9	424.1	531.9	668.7	1.5	54.8
	38	–	171.7	–	36,000	336.6	295.1	355.2	452.1	1.5	48.9
	39*	–	142.2	–	26,626	191.0	185.2	196.9	458.5	1.4	25.5
	40*	–	201.8	–	32,124	520.4	486.8	558.9	621.2	1.5	61.2
								Avg.	468.1		43.6
80	41	–	261.0	–	36,000	398.0	385.3	409.7	436.9	3.4	34.4
	42*	–	201.8	–	32,456	299.6	295.2	310.6	952.8	3.7	32.6
	43	–	295.2	–	36,000	424.4	415.9	435.3	972.7	4.1	30.4
	44	–	202.3	–	36,000	344.5	322.5	365.4	739.9	3.8	41.2
	45	–	286.7	–	36,000	524.6	502.3	552.9	924.9	3.9	45.3
	46	–	236.2	–	36,000	376.4	368.9	380.7	606.1	3.6	37.2
	47*	–	246.0	–	30,045	347.9	336.9	355.5	521.5	3.4	29.3
	48*	–	277.9	–	33,126	352.5	346.1	355.5	332.3	3.1	21.2
	49	–	323.9	–	36,000	619.9	584.0	638.4	754.5	2.9	47.7
	50*	–	297.5	–	11,632	792.2	744.7	835.8	888.4	3.0	62.4
								Avg.	713		38.2

larger terminals. The efficiency of berthing resources and the berth-on-arrival rates are Key Performance Indicators (KPIs) for large scale terminals. To improve these KPIs, larger terminals aim at utilizing such highly packed berth plans in the best way. Thus, it is important to evaluate new algorithms on such scenarios.

Table 12 holds the results of BACAP+ and ALNS for the new benchmark. The instances are indexed by following Table 9. The four columns under BACAP+ can be interpreted as in Table 8. The best performing settings of BACAP+ have been used when solving the models. For ALNS we report average, best and worst results of 10 runs along with the computational time of ALNS, the time to generate the initial assignments and the gap of average ALNS results (G_{ALNS} =(Average Cost-LB)/Average Cost). The ALNS is run with the suggested settings except for the 80 vessels instances. For these instances, the removal parameter ϕ_r is set to 10 in order to reduce the computational time. The computational results in Table 9 have shown that the ALNS performance is getting stronger as the size of instance increases. However, we cannot compare ALNS results with other state-of-the-art heuristics since this is a new benchmark.

The BACAP+ model cannot generate feasible solutions for the new instances within 10 h of runtime. For this reason, the performance of ALNS is evaluated on the LBs from the BACAP+. There are also eight instances where the solver runs out of memory. These are marked with "$*$", and the time when solver runs out of memory is also reported. The average relative gap of ALNS is 43.6% and 38.2%, and the average run time is 488.1 and 713 s for the 60 and 80 vessels instances, respectively. The relative gaps of the ALNS are comparatively high for the new benchmark. This is mainly because we evaluate the ALNS performance on BACAP+ LBs which might not be strong. Moreover, the new larger benchmark is computationally harder since it is more densely-packed. We believe that these results further motivate future research on exact methods for the BACAP. These results can be valuable when testing new heuristics.

7. Conclusions

In this paper, we have proposed a number of families of valid inequalities and variable fixing methods to improve the state-of-the-art BACAP formulations. Additionally, we have presented an ALNS heuristic to solve the problem more efficiently. The results show that both inequalities and the heuristic improve the state-of-the-art results. The inequalities and the ALNS can easily be used for related research problems with some modifications.

Various novel parameters have been efficiently used to formulate the inequalities. We should note that the average gaps of the BACAP+ model are consistently better than those of GSPP and BACAP models for each instance size. With respect to gaps, the BACAP+ model is better or at least the same for all, except six instances. The average gap has been reduced by 76% in average with the use of the enhancements for the BACAP model. The BACAP+ model obtains better (or at least finds the same) upper bounds for all, except one instance. The best LBs obtained by BACAP+ model are also competitive with the GSPP model of Iris et al. (2015).

The exact methods presented in this paper tend to require too much computational effort to be of practical use. In contrast, the proposed ALNS approach is able to provide high-quality (in average 3.1% from the best known upper bounds) results in short computational times. What is more, the computational time of ALNS remains to be less than 20 min even for the largest scale instances. This gives the opportunity to evaluate the berthing plan more frequently in the dynamic environment of a large container terminal, which allows the planner to more efficiently handle adjustments due to e.g. schedule changes or disruptions.

The ALNS heuristic uses different destroy and repair methods to diversify the search. Comparative tests on instances have shown that our heuristic can produce high-quality solutions compared to other heuristics presented in the state-of-the-art both on computational time and solution quality. We also generate a new larger benchmark suite for this problem and report computational results for these instances.

A natural topic for future research can be the assignment of specific QC. This could be achieved modifying the ALNS to solve the BACASP with the QC setup times and QC non-overtaking requirements. Another important extension of the problem definition is time-dependent berthing times due to tides (Lalla-Ruiz et al., 2016), speed optimization (Venturini et al., 2017).

An interesting research avenue is using the strengthened bounds of Section 4 for the ALNS heuristic. This can be applied when the repair operators are based on mathematical models, and one can utilize the strengthened bounds to improve the performance of the repair operators.

Acknowledgments

Authors would like to thank Frank Meisel (Kiel University) for contributing benchmark instances of the studied problem, and helpful comments. We also appreciate the comments and suggestions of four anonymous referees. This project was supported by The Danish Maritime Cluster which was funded by Vækstforum Hovedstaden and the European Social Fund (DTU research projects number: 35307). Stefan Ropke was supported by The Danish Strategical Research Council and The Danish Energy Technology Development and Demonstration Program (EUDP) under the ENERPLAN and GREENSHIP project.

Appendix A. Calculation of $\hat{q}, f_1, f_2$ (Iris et al., 2015)

Algorithm 4. QC assignment

```
   Input     : $k, r_k^{min}, r_k^{max}, process_i, (1+\beta\Delta b_i)m_k$
1  Find $\hat{q} \in \{r_k^{min}, ..., r_k^{max}\}$ such that $\hat{q} = \left\lfloor \left(\frac{(1+\beta\Delta b_i)m_k}{process_i}\right)^{1/\alpha} \right\rfloor$
2  $p = process_i$
3  while $(p \geq 0)$ do
4      $\delta = p(\hat{q}+1)^\alpha + (process_i - p)(\hat{q})^\alpha$
5      if $\delta \geq (1+\beta\Delta b_i)m_k$ then
6          $f_2 = p, f_1 = (process_i - p)$
7      $p = p - 1$
8  return $f_1, f_2, \hat{q}$
```

Appendix B. Lower bound on objective function: $\underline{z}^1$ (Iris et al., 2015)

We first obtain $f_1, f_2, \hat{q}$ by using Appendix A. We know that an assignment j has a $cost_j$ which is time-dependent cost component and QC assignment cost. With this information, we can calculate an improved lower bound $\phi(j)$ for assignment j's cost contribution to the objective function:

$$\phi(j) = cost_j + c_4(f_1\hat{q} + f_2(\hat{q}+1)) \quad \text{(B.1)}$$

We now use $\phi(j)$ to define the lowest contribution σ_i for each vessel i:

$$\sigma_i = \min_{j \in \Omega_i}\{\phi(j)\} \tag{B.2}$$

and we compute the lower bound on the total objective: $\underline{z}^1 = \sum_{i \in V} \sigma_i$.

References

Bierwirth, Christian, Meisel, Frank, 2015. A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. Eur. J. Oper. Res. 244 (3), 675–689.

Buhrkal, Katja, Zuglian, Sara, Ropke, Stefan, Larsen, Jesper, Lusby, Richard, 2011. Models for the discrete berth allocation problem: a computational comparison. Transport. Res. Part E: Logist. Transport. Rev. 47 (4), 461–473.

Carlo, Héctor J., Vis, Iris F.A., Roodbergen, Kees Jan, 2015. Seaside operations in container terminals: literature overview, trends, and research directions. Flexible Services Manuf. J. 27 (2), 224–262.

Coelho, Leandro C., Cordeau, Jean François, Laporte, Gilbert, 2012. The inventory-routing problem with transshipment. Comput. Oper. Res. 39, 2537–2548.

Cordeau, Jean-François, Laporte, Gilbert, Legato, Pasquale, Moccia, Luigi, 2005. Models and Tabu search heuristics for the berth-allocation problem. Transport. Sci. 39 (4), 526–538.

Cordeau, Jean François, Laporte, Gilbert, Moccia, Luigi, Sorrentino, Gregorio, 2011. Optimizing yard assignment in an automotive transshipment terminal. Eur. J. Oper. Res. 215, 149–160.

Giallombardo, Giovanni, Moccia, Luigi, Salani, Matteo, Vacca, Ilaria, 2010. Modeling and solving the tactical berth allocation problem. Transport. Res. Part B: Methodol. 44 (2), 232–245.

Golias, Mihalis, Portal, Isabel, Konur, Dinçer, Kaisar, Evangelos, Kolomvos, Georgios, 2014. Robust berth scheduling at marine container terminals via hierarchical optimization. Comput. Oper. Res. 41, 412–422.

Guan, Yongpei, Cheung, Raymond K., 2004. The berth allocation problem: models and solution methods. OR Spectrum 26 (1), 75–92.

Hansen, Pierre, Oguz, Ceyda, Mladenovic, Nenad, 2008. Variable neighborhood search for minimum cost berth allocation. Eur. J. Oper. Res. 191 (3), 636–649. ISSN: 0377-2217.

He, Junliang, 2016. Berth allocation and quay crane assignment in a container terminal for the trade-off between time-saving and energy-saving. Adv. Eng. Inform. 30 (3), 390–405.

Hsu, Hsien-Pin, 2016. A hpso for solving dynamic and discrete berth allocation problem and dynamic quay crane assignment problem simultaneously. Swarm Evol. Comput. 27, 156–168.

Imai, Akio, Nishimura, Etsuko, Papadimitriou, Stratos, 2001. The dynamic berth allocation problem for a container port. Transport. Res. Part B: Methodol. 35 (4), 401–417.

Imai, Akio, Sun, Xin, Nishimura, Etsuko, Papadimitriou, Stratos, 2005. Berth allocation in a container port: using a continuous location space approach. Transport. Res. Part B: Methodol. 39 (3), 199–221.

Imai, Akio, Chen, Hsieh Chia, Nishimura, Etsuko, Papadimitriou, Stratos, 2008. The simultaneous berth and quay crane allocation problem. Transport. Res. Part E: Logist. Transport. Rev. 44 (5), 900–920.

Imai, Akio, Yamakawa, Yukiko, Huang, Kuancheng, 2014. The strategic berth template problem. Transport. Res. Part E: Logist. Transport. Rev. 72, 77–100.

Iris, Çağatay, Pacino, Dario, Ropke, Stefan, Larsen, Allan, 2015. Integrated berth allocation and quay crane assignment problem: set partitioning models and computational results. Transport. Res. Part E: Logist. Transport. Rev. 81, 75–97.

Jin, Jian Gang, Lee, Der-Horng, Hu, Hao, 2015. Tactical berth and yard template design at container transshipment terminals: a column generation based approach. Transport. Res. Part E: Logist. Transport. Rev. 73, 168–184.

Kordic, Stevan, Davidovic, Tatjana, Kovac, Natasa, Dragovic, Branislav, 2016. Combinatorial approach to exactly solving discrete and hybrid berth allocation problem. Appl. Math. Model. 40 (21–22), 8952–8973.

Lalla-Ruiz, Eduardo, González-Velarde, José Luis, Melián-Batista, Belén, Marcos Moreno-Vega, J., 2014. Biased random key genetic algorithm for the tactical berth allocation problem. Appl. Soft Comput. J. 22, 60–76.

Lalla-Ruiz, Eduardo, Expósito-Izquierdo, Christopher, Melián-Batista, Belén, Marcos Moreno-Vega, J., 2016. A set-partitioning-based model for the berth allocation problem under time-dependent limitations. Eur. J. Oper. Res. 250 (3), 1001–1012.

Lee, Der Horng, Chen, Jiang Hang, Cao, Jin Xin, 2010. The continuous berth allocation problem: a greedy randomized adaptive search solution. Transport. Res. Part E: Logist. Transport. Rev. 46 (6), 1017–1029.

Li, Feng, Sheu, Jiuh-Biing, Gao, Zi-You, 2015. Solving the continuous berth allocation and specific quay crane assignment problems with quay crane coverage range. Transport. Sci. 49 (4), 968–989.

Liang, Chengji, Huang, Youfang, Yang, Yang, 2009. A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning. Comput. Indust. Eng. 56 (3), 1021–1028.

Liu, Jiyin, Wan, Yat-wah, Wang, Lei, 2006. Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. Naval Res. Logist. 53 (1), 60–74.

Mauri, Geraldo Regis, Ribeiro, Glaydston Mattos, Lorena, Luiz Antonio Nogueira, Laporte, Gilbert, 2016. An adaptive large neighborhood search for the discrete and continuous berth allocation problem. Comput. Oper. Res. 70, 140–154.

Meisel, Frank, Bierwirth, Christian, 2009. Heuristics for the integration of crane productivity in the berth allocation problem. Transport. Res. Part E: Logist. Transport. Rev. 45 (1), 196–209.

Meisel, Frank, Bierwirth, Christian, 2013. A Framework for integrated berth allocation and crane operations planning in seaport container terminals. Transport. Sci. 47 (2), 131–147.

Muller, Laurent Flindt, Spoorendonk, Simon, Pisinger, David, 2012. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. Eur. J. Oper. Res. 218 (3), 614–623.

Park, Young-Man, Kim, Kap Hwan, 2003. A scheduling method for berth and quay cranes. OR Spectrum 25 (1), 1–23.

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

Rodriguez-Molins, Mario, Salido, Miguel a., Barber, Federico, 2014. A GRASP-based metaheuristic for the berth allocation problem and the quay crane assignment problem by managing vessel cargo holds. Appl. Intell. 40, 273–290.

Ropke, Stefan, Pisinger, David, 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transport. Sci. 40 (December), 455–472.

Shang, Xiao Ting, Cao, Jin Xin, Ren, Jie, 2016. A robust optimization approach to the integrated berth allocation and quay crane assignment problem. Transport. Res. Part E: Logist. Transport. Rev. 94, 44–65. ISSN: 1366-5545.

Shaw, Paul, 1998. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. Springer, Berlin, Heidelberg, pp. 417–431, ISBN: 978-3-540-49481-2.

Stahlbock, Robert, Voß, Stefan, 2007. Operations research at container terminals: a literature update. OR Spectrum 30 (1), 1–52.

Turkogullari, Yavuz B., Caner Taskin, Z., Aras, Necati, Kuban Altinel, I., 2014. Optimal berth allocation and time-invariant quay crane assignment in container terminals. Eur. J. Oper. Res. 235 (1), 88–101.

Turkogullari, Yavuz B., Caner Taskin, Z., Aras, Necati, Kuban Altinel, I., 2016. Optimal berth allocation, time-variant quay crane assignment and scheduling with crane setups in container terminals. Eur. J. Oper. Res. 254 (3), 985–1001.

UNCTAD, 2015. Review of Maritime Transport: 2015. ISBN: 9789211128109.

Ursavas, Evrim, 2014. A decision support system for quayside operations in a container terminal. Decis. Support Syst. 59, 312–324.

Vacca, Ilaria, Salani, Matteo, Bierlaire, Michel, 2013. An exact algorithm for the integrated planning of berth allocation and quay crane assignment. Transport. Sci. 47 (2), 148–161.

Venturini, Giada, Iris, Çağatay, Kontovas, Christos A., Larsen, Allan, 2017. The multi-port berth allocation problem with speed optimization and emission considerations. Transport. Res. Part D: Transport Environ. 54, 142–159.

Zhen, Lu, Chew, Ek Peng, Lee, Loo Hay, 2011. An integrated model for berth template and yard template planning in transshipment hubs. Transport. Sci. 45 (4), 483–504.