Flexible ship loading problem with transfer vehicle assignment and scheduling

Çağatay Iris[a,b,*], Jonas Christensen[a], 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*

Abstract

This paper presents the flexible containership loading problem for seaport container terminals. The integrated management of loading operations, planning of the transport vehicles to use and their scheduling is what we define as the Flexible Ship Loading Problem (FSLP). The flexibility comes from a cooperative agreement between the terminal operator and the liner shipping company, specifying that the terminal has the right to decide which specific container to load for each slot obeying the class-based stowage plan received from the liner. We formulate a mathematical model for the problem. Then we present various modelling enhancements and a mathematical model to obtain strong lower bounds. We also propose a heuristic algorithm to solve the problem. It is shown that enhancements improve the performance of formulation significantly, and the heuristic efficiently generates high-quality solutions. Results also point out that substantial cost savings can be achieved by integrating the ship loading operations.

1. Introduction

Maritime freight transport constitutes an important part of the global logistics systems. Benefiting from rapid globalisation, the containerised freight transport has been steadily growing over the past decade apart from the year 2009 with the global financial crisis. The leading 100 container terminals handled 539.2 million Twenty Equivalent Units (TEUs) in 2015 (UNCTAD (2016)) with an increase by 6.8% from 2014. Therefore, the increasing container handling volumes make operations planning a more complex and significant challenge for container terminals.

Liner shipping companies have adapted to the growth in the transport volumes by increasing the capacity of their services. This is done by deploying mega vessels of over 20,000 TEUs and planning more frequent visits to the container terminals. Capacity is, however, not enough. A reliable shipping service ensures that the cargoes arrive on time, so container terminals are required to supply reliable and agile operations for their customers. The increase in vessels size intensifies the pressure on the container terminals. Meanwhile, shipping companies also expect terminals to minimise the vessel turnaround (handling) times.

Vessel turnaround times might be reduced by deploying more Quay Cranes (QCs) and Transfer Vehicles (TVs) on each vessel, however, this does not guarantee an improvement in the service quality. There is a limited number of equipment that can be assigned to a vessel. Also, inefficient management of this equipment can bring more congestion and deterioration in the overall performance. Considering that QCs and TVs are limited resources with high operating costs, terminals should rather optimise the use of these resources.

We refer readers to the literature reviews on decision problems in seaside operations (Carlo et al. (2013), Bierwirth and Meisel (2015)), transport operations (Steenken et al. (2004), Carlo et al. (2014b)) and yard operations (Li and Vairaktarakis (2004), Carlo et al. (2014a)) in terminals. Literature reviews such as Kim and Lee (2015) note that there is a need for flexibility in operations, and possible collaboration with the liner shipping company can bring some flexibility in the ship loading related operations.

The efficient loading of containers to the vessel has become a more complicated problem due to the increase in vessel size, vessel numbers and complex technicalities. The high degree of industrial requirements (e.g. lashing patterns, vessel stress forces and staff working hour regulations) along with all other mentioned challenges, make efficient ship loading an even more complicated problem. It also often happens that some of the containers are ready to be loaded earlier but have to wait since they would be out of the planned load sequence. Due to the mentioned complexities and limited handling equipment, most attempts to improve the loading operations could benefit from optimisation methods.

Some liner shipping companies are aware of the challenges that container terminals face and have actively started to adapt their *stowage plans* in such a way that gives the terminal operator more freedom to optimise the usage of their equipment. A stowage plan describes the arrangement of containers on the vessel. In recent years, there has been a shift in the stowage planning policy which is based on increasing collaboration between the terminal and the liner shipper. The

*Corresponding author
Email address: cagai@dtu.dk (Çağatay Iris)

liner provides the terminal with the stowage plan based on container classes (a container class is defined by the port of discharge, physical container dimensions, weight, etc.) which we refer to as *class-based stowage plan*. The terminal has the flexibility of determining the position of specific containers of the same class obeying the class-based stowage plan, and this ensures the flexible loading operations (Monaco et al. (2014)). In this study, we integrate the assignment and scheduling of transfer vehicles and container load sequencing with the assignment of specific containers to the vessel positions. We call the entire problem the Flexible Ship Loading Problem (FSLP). We aim at reducing service times of the handling equipment and meeting the deadlines on the finishing time of the loading.

The contribution of the study is multi-fold. First, we introduce a new integrated container terminal problem to improve the efficiency of the loading operations. The problem addresses many realistic and important aspects of the loading operations. We formulate a mathematical model to solve the problem and some enhancements to improve this formulation. Then we suggest a model to obtain lower bounds for the problem. We also propose a heuristic method to solve it. Computational results show that the enhancements on the model significantly improve its performance, but still, the mathematical model is intractable for large-scale instances. The results for the heuristic show that it outperforms the mathematical model with respect to both solution quality and computation time, and instances, with up to 1000 containers to be loaded, are solved very efficiently. We also show that there are significant cost savings by integrating these problems rather than solving them in a hierarchical manner.

The remainder of the paper is organised as follows. Section 2 briefly presents relevant literature. Section 3 includes the problem definition. Section 4 provides the mathematical model and enhancements on this formulation, while Section 5 presents a new method to obtain lower bounds. The heuristic is detailed in Section 6. The results are discussed in Section 7, and finally, the conclusions and future research perspectives are presented in the last section.

2. Relevant literature

The problem studied in this paper is related to the ship loading operations, and it covers aspects such as stowage planning, load sequencing, and handling equipment routing and scheduling. A detailed literature review on all of these problems can be found in Iris and Pacino (2015).

The stowage planning problem has been addressed in two different ways in the literature. Some papers aim at minimising handling costs ensuring stability and seaworthiness of a ship in its route containing multiple ports. These studies agree that the problem belongs to the liner shipping company (see Pacino et al. (2011), Parreno et al. (2016)). In this paper, we review studies that consider the stowage planning problem for a single container terminal. Imai et al. (2002) study the stowage planning at a single terminal with the aim of minimising yard re-handles and the stability measure GM (i.e. the distance between the centre of gravity and the metacentre). Imai et al. (2002) call this problem the containership loading problem. In comparison to Imai et al. (2002), our work ensures that the stowage plan satisfies a class-based stowage plan that comes from the carrier. Later, Imai et al. (2006) include trim and heeling to the objective function, and they also extend the problem by covering multiple rows in the yard. In Ambrosino and Sciomachen (2003), a stowage planning problem is solved in the first stage, and then two yard-handling strategies are evaluated with the suggested stowage plan. The vessel stability is addressed by balancing the front-back and right-left side of the ship (the details of the stowage planning problem are in Ambrosino et al. (2004)). Recently, Monaco et al. (2014) distinguish between the stowage planning problem solved by the liner shipping company (resulting in a class-based stowage plan) and the specific container assignment problem of the terminal (called *operational stowage planning problem*). They solely study the operational stowage planning problem and solve it through a two-phase tabu search method. None of the above studies integrates the planning of the yard transport equipment with the stowage planning which is subject of investigation in our paper. For such integrated planning, Steenken et al. (2001) is the pioneering work. In comparison to our work, Steenken et al. (2001) approach the problem with a just-in-time method. They solve a model that assigns each container to a specific position and a specific Straddle Carrier (SC) for only one QC. In the case when multiple QCs serve the ship simultaneously, they do not solve the specific container allocation problem for all positions simultaneously unlike this paper. Instead, they suggest a best-fit method for each QC.

The second component of the FSLP is the load sequencing problem in which the loading order of containers is decided. Such a problem is directly attached to QC assignment and scheduling problem for a single ship (See examples such as Kim and Park (2004), Legato et al. (2012)) where required QCs are assigned to each set of bays, and the loading order of the bays and positions are determined for each QC. The load sequencing problem, which also determines the retrieval order from the yard, is also related to the locations of the containers in the yard (See papers that determine yard location, e.g. Jiang and Jin (2017), Zhen et al. (2016b)). Ji et al. (2015) study the load sequencing problem with multiple QCs and yard configurations. The authors suggest three different container relocation strategies which determine the relocation position of the blocking containers in the yard. They solve the problem with a Genetic Algorithm (GA) based method. Bian et al. (2016) determine the loading sequence considering the number of re-handles in the yard. Bian et al. (2016) assume that a detailed stowage plan is given and that only one QC is available to load the ship. They suggest a two-phase method where they first order containers which do not require any re-handling, and then use a dynamic programming algorithm to sequence the remaining containers. Kim et al. (2004) integrate the load sequencing and Yard Truck (YT)

scheduling considering a given class-based stowage plan. The vessel stability is ensured by imposing weight and height limit constraints on the stacks. The load sequencing is optimised, but a column-wise loading policy is prioritised. A two-stage method is suggested. In the first stage, yard-clusters are sequenced (as in Lee et al. (2005)), while specific containers are sequenced in the second stage. Some studies address the integrated load sequencing and QC planning problems including internal reshuffles within a bay of the vessel (e.g. Meisel and Wichmann (2010), Ding et al. (2017)).

Carlo et al. (2014b) point out that there are many papers about the assignment of the transport equipment to QCs and/or containers (e.g. Bish et al. (2005)), and the scheduling and routing of the equipment (e.g. Kim and Kim (1999), Zeng and Yang (2009)). In this paper, we solely review studies that integrate the handling equipment planning with load sequencing and/or stowage planning. Alvarez (2006) integrate reach-stackers scheduling with stowage planning. The paper assumes that loading policy is either column-wise or layer-by-layer, and it is decided beforehand. The problem aims at minimising the number of re-handling, the travelling distance and vessel instability. A Tabu Search (TS) based solution method is suggested. In a later work (Alvarez (2008)), a Lagrangian Relaxation (LR) based solution approach has been suggested for a similar problem. Jung and Kim (2006) integrate the load scheduling with yard crane scheduling for a single ship. The authors consider the interference between adjacent yard cranes and attempt to minimise makespan.

3. The Flexible Ship Loading Problem

The FSLP is the integration of four planning problems in the terminal, namely operational stowage planning, load sequencing, equipment assignment and equipment scheduling. This problem is configured to load a single ship, but it could be extended to multiple ships.

We now detail the FSLP by describing each of its components and their interactions. The operational stowage planning problem (Monaco et al. (2014)) deals with assigning specific containers in the yard to one slot (i.e. position) on the vessel with respect to the class-based stowage plan supplied by liner shipping company. In Figure 1, a class-based stowage plan is illustrated for a bay of a ship. In this figure, there are four container classes to be loaded into the bay, namely A, C, D, E, and the positions of containers in the yard area are shown. Two containers of the same class have the same dimension and type, the same discharge port and a similar weight. The weight difference is small enough so it does not have a huge impact on the stability of the vessel. The liner shipping company is responsible for making the class-based stowage plan while ensuring the vessel is seaworthy. The terminal then has the freedom to make a more detailed operational stowage plan, where each position is assigned a specific container of the correct class. The number of vessel positions for a class should exactly match to the number of containers of that class to be loaded. While making the operational stowage plan, the terminal can take into account the planning of in-house equipment, and thus optimise the operations. Figure 1 is a simplified example of one bay and yard, while the problem is solved for all bays that will be loaded.

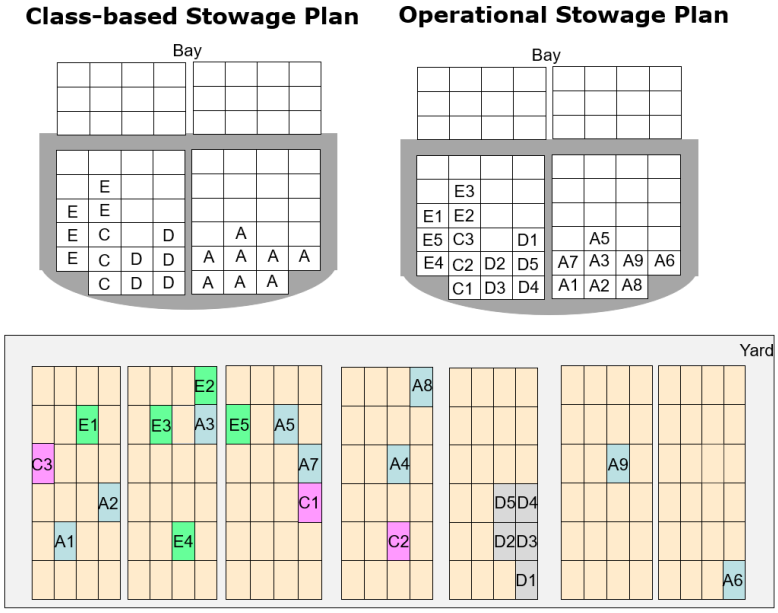


Figure 1: Class-based stowage plan to operational stowage plan

The flexibility of selecting the specific container (among the same class containers) might bring significant savings on the travelling distance, and consequently the travelling time, of all containers. Let us consider the example in Figure 2 with two containers of the same class (say two 40-foot containers weighing between 20 and 22 tonnes having the same destination) are to be loaded. The position of each container in the yard area is also shown, and an arrow represents the

travelling distance needed to bring the container to the vessel. In the figure on the right side, the terminal has made a better operational stowage plan that requires less travelling distance compared to the operational stowage plan on the left side. Besides the minimisation of the travelling time, scheduling vehicles also provide an overview of the tasks to be performed. Thus, if a container is far away from the QC, terminal can account for this when doing the scheduling, and start to retrieve it considering the travelling distance. Doing so, terminal can avoid waiting time for the QC.

The flexible assignment of containers in the same class helps to integrate the remaining problems into the FSLP. The operational stowage plan is of no consequence for the liner so long as the container classes are not changed.

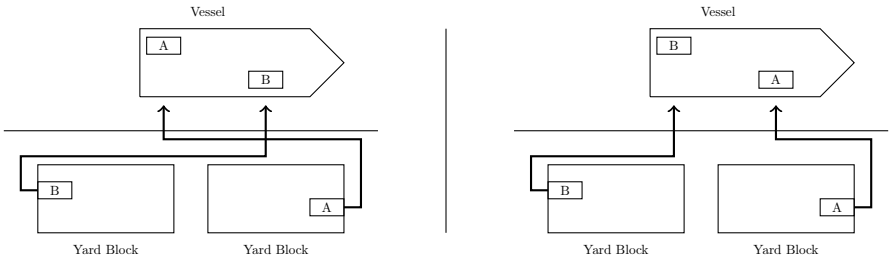


Figure 2: Different operational stowage plans: Effect on travel distance.

The second component of the FSLP is the load sequencing problem which determines the loading sequence of the containers in the yard. The sequence in which containers will be loaded is governed by physical rules, and due to the specific layout of the vessel, some positions must be loaded before others. The sequence is affected by the ready time (i.e. the time a container is in front of the respective QCs) of each container, and this depends on various factors such as container locations in the yard, the availability of TVs, the operational stowage plan, etc. The terminal might reduce the total loading time by efficiently sequencing containers (See Kim et al. (2004)).

The load sequencing problem is constrained by the QCs work-schedule. The QCs work-schedule is a set of decisions that include the QCs assignment to bays of the vessel and the loading order between the bays. The QC work-schedule is mostly determined in earlier stages with berth allocation and QC assignment problem (See Iris et al. (2015), Turkogullari et al. (2016), Iris et al. (2017)).

The FSLP finally covers the assignment and scheduling of transport equipment (i.e. transfer vehicles). Integrating the TVs into the FSLP is vital because they are limited resources in the terminal, and they might cause a bottleneck during loading operations. Moreover, generating a feasible schedule of TVs will determine the ready time of each container in front of the respective QC more accurately. These ready times influence the assignments of containers to each position (operational stowage plan). In this study, time is discretised by minutes, and the time unit is one minute. The FSLP studied in this paper covers the assignment of specific TVs to each QC and the scheduling of all TVs to load all containers. In other words, the problem deals with determining which specific container will be picked up by which TV at what point in the time. Each TV is assigned to a specific QC for the entire planning horizon, but the TV does not necessarily have to work from start to finish. This means that, for example, a solution can hold 3 TVs working on a QC for some time then it can be reduced to 2 TVs for the remaining loading time. We consider such a time-variant TV assignment in this study.

The problem definition is based on the following assumptions:

- Unloading operations are performed first, then loading operations start (i.e. the problem does not include double cycling (Goodchild and Daganzo (2007))).
- QCs have been assigned to bays on the vessel beforehand, and each QC's loading order is known. The load sequencing policy, "from-sea-to-land" (to ensure adequate visibility for QC operator as noted in Steenken et al. (2001)) with "stack-wise" sequencing is applied by each QC.
- Each container in the yard is ready to be retrieved when the transfer vehicle arrives at the respective yard bay to pick up the container (e.g. a pre-marshalling problem has been solved beforehand).
- The stability of the vessel is ensured with the class-based stowage-plan. Note that Pacino et al. (2011) have shown that the stability errors in class-based stowage plans are mostly negligible.
- Each TV can only work for a single QC during the loading of the vessel. In other words, it is not allowed to pool TVs for QCs, and all TVs are identical.
- TV operations are non-preemptive. When a TV is assigned to a QC, it does not stop until it finishes the given tasks on that QC, and each TV is in front of its respective QC in the initial position.
- The congestion in the yard, the travel speed of a TV with/without a container are all reflected in the transportation times between yard positions (or Input/Output points depending on the yard layout type) and QCs.

- There is no container buffer area under the QC, and this means the TV and QC operations are not decoupled.

In the FSLP we only optimise the loading operations. The main reason for this is that there is less flexibility to exploit on the discharge operations since the unloading order and yard destination of the unloaded containers are mostly determined beforehand.

A feasible solution to the FSLP holds the assignment of each container to a vessel position. A solution also shows which TV will transfer each container to its position, the time of pickup from the yard block and delivery time in front of the QC. The complete schedule of all TVs is also made. A feasible solution fulfils the requirements of the class-based stowage plan, QC work-schedules and the load sequencing policy. We believe that our study is the first to integrate the above four problems. It utilises the collaboration between liner shipping company and terminal operators.

4. Mathematical Model for the FSLP

4.1. FSLP Model

The list of notations, i.e. parameters, decision variables, is as follows:

Parameters and sets:	
C	Set of containers that will be loaded to vessel
C_p	Set of containers belonging to a class suitable for slot position p
Q	Set of quay cranes that are assigned to load the vessel
Q_p	Set of quay crane that loads position p (one element set)
P	Set of positions on the vessel to be loaded
P_i^{con}	Set of positions on the vessel that match with the class type of container i
P_q	Set of positions on the vessel that will be loaded by QC q
P_p^{crane}	Set of positions on the vessel that are handled by the same crane as position p
S	Set of transfer vehicles available to serve the vessel
S_p^{pos}	Set of transfer vehicles that are available to serve the vessel-position p
S_q	Set of transfer vehicles that are available to serve QC q, $S_q \in \left\{s_q^1, s_q^2, .., s_q^{\lvert S_q \rvert}\right\}$
T	Set of time periods, $T \in \{0, 1, .., H-1\}$, where H is the closing of planning horizon
τ_{ip}	Time needed for a transfer vehicle to transport container i from its yard-position to vessel-position p. The time needed is assumed to be equal in both directions.
β	The loading time for each QC, minimum time between two consecutive container loading operations
EFT	Expected finishing time of operations for the vessel
α	The cost of using one TV for one time-unit
γ	The cost of exceeding the expected finishing time (EFT) for one time-unit
M	A large positive number
Decision variables:	
$t_p^s \in \mathbb{Z}^+$	Time when container for position p has been dropped in front of QC by TV s (container dropping time). If a position p is not served by TV s, then $t_p^s = 0$.
$Start_s \in \mathbb{Z}^+$	Time when operations of TV s start
$End_s \in \mathbb{Z}^+$	Time when operations of TV s end
$z \in \mathbb{Z}^+$	Makespan for the loading of entire ship (operations)
$\Delta EFT \in \mathbb{Z}^+$	The maximum tardiness of operations, i.e. the number of time units the loading is finished after the EFT. ΔEFT is 0 if the operations are finished at, or before the EFT.
$x_{ip}^s \in \mathbb{B}$	1; if the container i is loaded to position p, and it is picked up by TV s, 0 otherwise

As per the second assumption mentioned in Section 3, the load ordering of the positions on the vessel is determined beforehand. We use the notation $\bar{p} \prec\prec p$ to indicate that the position $\bar{p}$ is handled immediately before position p according to the ordering, while $p' \prec p$ indicates that position p' is loaded before position p, by the same QC.

The binary decision variables x_{ip}^s correspond to the operational stowage plan and TV assignment to containers, while integer variables $t_p^s, Start_s, End_s$ handle the TV scheduling. Let us now introduce the mathematical model:

$$\min \alpha \sum_{s \in S} (End_s - Start_s) + \gamma \Delta EFT \tag{1}$$

subject to

$$\sum_{p \in P_i^{con}} \sum_{s \in S_p^{pos}} x_{ip}^s = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{i\in C_p}\sum_{s\in S_p^{pos}} x_{ip}^s = 1 \qquad \forall p \in P \tag{3}$$

$$2\tau_{ip} - M(2 - x_{ip}^s - \sum_{i\prime\in C_{p\prime}} x_{i\prime p\prime}^s) \le t_p^s - t_{p\prime}^s \qquad \forall i \in C, \forall s \in S, \forall p \in P_i^{con}, p\prime \in P_p^{crane} \mid p\prime \prec p \tag{4}$$

$$\sum_{s\in S} t_p^s \ge \sum_{s\in S} t_{\bar{p}}^s + \beta \qquad \forall p \in P, \forall \bar{p} \in P_p^{crane} \mid \bar{p} \prec\prec p \tag{5}$$

$$t_p^s \le \sum_{i\in C_p} H x_{ip}^s \qquad \forall p \in P, \forall s \in S_p^{pos} \tag{6}$$

$$t_p^s \ge 2\sum_{i\in C} \tau_{ip} x_{ip}^s \qquad \forall p \in P, \forall s \in S_p^{pos} \tag{7}$$

$$t_p^s - 2\sum_{i\in C} \tau_{ip} x_{ip}^s + H(1 - \sum_{i\in C} x_{ip}^s) \ge Start_s \qquad \forall s \in S, \forall p \in P \tag{8}$$

$$t_p^s \le End_s \qquad \forall s \in S, \forall p \in P \tag{9}$$

$$Start_s \le End_s \qquad \forall s \in S \tag{10}$$

$$z \ge t_p^s + \beta \qquad \forall s \in S, \forall p \in P \tag{11}$$

$$x_{ip}^s = 0 \qquad \forall i \in C, \forall p \in P, \forall s \in S \setminus S_p^{pos} \tag{12}$$

$$\Delta EFT \ge z - EFT \tag{13}$$

$$t_p^s, Start_s, End_s, z, \Delta EFT \in \{0, .., H-1\} \qquad \forall s \in S, \forall p \in P \tag{14}$$

$$x_{ip}^s \in \{0, 1\} \qquad \forall i \in C, \forall s \in S, \forall p \in P \tag{15}$$

The objective function (1) is a combination of the cost of service times of TVs and the maximum tardiness (if the loading finishes after the expected finishing time). Constraint (2) ensures that each container will be loaded into a position that matches with its container class. Constraint (3) guarantees that all positions are loaded with a container that matches the container class of that position. For a given container, TV and position, constraint (4) makes sure that the container dropping time for that position is set correctly. This is done by considering two positions that will be loaded by the same QC, and forcing the difference between their dropping times to be greater than or equal to the time required to bring the container ($2\tau_{ip}$) in front of the QC. The term multiplied by M on the left-hand side in constraint (4) makes sure that the constraint is only active when the two positions are transported by the same TV. Constraint (5) ensures that all positions are loaded in the correct order, and the containers that will arrive at the same QC should have at least β time apart. Constraint (6) ensures that $t_p^s = 0$ if TV s does not serve position p. Constraint (7) guarantees that the earliest dropping time for a position is the transportation time of the container which is loaded to that position. Constraint (8) sets the starting time of each TV operation, while constraint (9) sets the ending time of each TV operation. If a TV is not assigned to any position, these variables take a value of zero. Constraint (10) is the link between the starting and ending time for each TV operation. Constraint (11) obtains the makespan, while constraint (12) ensures that a container for position p cannot be picked up by TV s if TV s is not assigned to serve the QC for position p. Constraint (13) calculates the maximum tardiness. Constraints (14)-(15) determine the domain of variables. Firstly, we formulate an upper bound on H simply by assuming that only one QC and one TV are used. We can, thus, obtain an upper bound on the planning horizon $H = \max\Big\{\sum_{p\in P}\Big\{2\max_{c\in C_p}\{\tau_{cp}\} + \max\{0, (\beta - 2\min_{c\in C_p}\{\tau_{cp}\})\}\Big\}, \beta|P|\Big\}$.

4.2. Enhancements for the FSLP model

This section introduces enhancements for our formulation. The enhancements are based on formulating lower bounds on variables and valid inequalities for the FSLP.

4.2.1. Lower bounds on the variables

Let us first formulate the minimum total transportation time that is required to transport containers that will be loaded by QC q. The minimum overall time needed to transport all containers of a particular QC q (δ_{min}^q) is obtained by solving an assignment problem that minimizes the total required transportation time. Let x_{cp} be the assignment variable, i.e. $x_{cp} = 1$ if container $c \in C$ is assigned position $p \in P$, and 0 otherwise. The value of δ_{min}^q can be calculated by solving the model in (16). The objective function minimises the transportation time obeying the class-based stowage plan, and the first constraint ensures that all positions that will be loaded by QC q must get exactly one container, and the second constraint ensures that each container can be loaded to at most one position.

$$\delta_{min}^q = \min\left\{\sum_{c\in C}\sum_{p\in P_c^{con}} 2\tau_{cp}x_{cp} : \sum_{c\in C_p} x_{cp} = 1 \;\; \forall p \in P_q, \sum_{p\in P_c^{con}} x_{cp} \le 1 \;\; \forall c \in C\right\} \qquad \forall q \in Q \tag{16}$$

We now set a lower bound on the completion time (i.e. makespan) variable z in constraint (17). The makespan must be greater than or equal to the finishing time of every QC. A lower bound on the finishing time of each QC is obtained by taking the maximum between the total loading time for QC q and the minimum transportation time needed to load all containers of QC q.

$$z \geq \max_{q \in Q} \left\{ \max\{\beta |P_q|, \left\lceil \frac{\delta^q_{min}}{|S_q|} \right\rceil \} \right\} \tag{17}$$

4.2.2. Valid inequalities for the FSLP model

We formulate a better link between t^s_p and x^s_{cp} variables in constraint (18). For each position p, the sum of dropping times (t^s_p) for all TVs is at least the maximum of the total loading time of all positions before p and the minimum transportation time required to load all positions before p.

$$\sum_{s \in S^{pos}_p} t^s_p \geq \max \left\{ \sum_{p' \prec p} \beta, \frac{\sum_{p' \prec p} \sum_{c \in C_{p'}} \sum_{s \in S^{pos}_{p'}} 2\tau_{cp'} x^s_{cp'}}{|S_{Q_p}|} \right\} \quad \forall p \in P \tag{18}$$

In constraint (18), the total loading time for all positions loaded by same QC before p is $\sum_{p' \prec p} \beta$. To calculate the minimum time to transport all containers before p, we first sum all transportation times for positions before p ($\sum_{p' \prec p} \sum_{c \in C_{p'}} \sum_{s \in S^{pos}_{p'}} 2\tau_{cp'} x^s_{cp'}$), we then divide this by total number of TVs assigned to that QC. This results in minimum transportation time for all positions before p. The two parts in constraint (18) are split up into two sets of constraints, to ensure the linearity of the model.

The next set of valid inequalities focuses on the container classes rather than specific containers. Let us name the set of container classes as U and the set of container classes for each QC q as U_q. These sets can be easily obtained as we know the QC work-schedules and the class-based stowage plan. We, also, can obtain the set of containers which is in the class of u (C_u), and the set of positions which requires a container of class u and which will be loaded by QC q (P_{qu}). Valid inequality (19) ensures that for each QC q and container class u belonging to set U_q, the total number of containers of class u to be loaded by QC q equals to $|P_{qu}|$.

$$\sum_{s \in S_q} \sum_{i \in C_u} \sum_{p \in P_{qu}} x^s_{ip} = |P_{qu}| \quad \forall q \in Q, \forall u \in U_q \tag{19}$$

We also formulate inequalities to break some symmetries. In this paper, it is assumed that all TVs available for a QC are identical. Hence a generated pickup order for a TV is feasible for all TVs. This property generates many symmetrical solutions where there are a lot of exactly indifferent alternative TV assignments. For each QC, constraint (20) ensures that for two TVs serving same QC, the one with the higher index cannot work for a longer time compared to the one with the lower index.

$$End_s - Start_s \geq End_{s+1} - Start_{s+1} \quad \forall q \in Q, \forall s \in S_q \setminus \{s_q^{|S_q|}\} \tag{20}$$

Finally, we formulate a valid inequality that better links the assignment (x^s_{cp}) and scheduling variables ($End_s, Start_s$). Constraint (21) ensures that the service time of TV s ($End_s - Start_s$) is greater than or equal to the total transportation time for the containers that are going to be handled by that TV s.

$$End_s - Start_s \geq 2 \sum_{i \in C_p} \sum_{p \in P^{con}_i} \tau_{ip} x^s_{ip} \quad \forall s \in S \tag{21}$$

We call the enhanced version of the FSLP model as FSLP+.

5. New lower bounds for the FSLP

To obtain new lower bounds for the FSLP, we focus on the components of the objective function, which are the cost of TV service times and the cost of ending later than expected finishing time. We formulate a new mathematical model that omits decision variables related to TV scheduling ($t^s_p, Start_s, End_s$), and this model obtains a lower bound for the FSLP. Let us first show that we can obtain lower bounds on each objective component by solely using x^s_{ip} variables.

Proposition 1: $\sum_{i \in C} \sum_{p \in P^{con}_i} \sum_{s \in S^{pos}_p} 2\tau_{ip} x^s_{ip}$ is a lower bound on $\sum_{s \in S} (End_s - Start_s)$.

Proof: $\sum_{i \in C} \sum_{p \in P^{con}_i} \sum_{s \in S^{pos}_p} 2\tau_{ip} x^s_{ip}$ constitutes the total transportation time of all TVs, while $\sum_{s \in S} (End_s - Start_s)$ is the service time, and it includes the total transportation time and the TV waiting times. Then, $\sum_{i \in C} \sum_{p \in P^{con}_i} \sum_{s \in S^{pos}_p} 2\tau_{ip} x^s_{ip} \leq$

$\sum_{s\in S}(End_s - Start_s)$ as the waiting time is always non-negative. □

Proposition 2: $\beta + \max_{s\in S}\Big\{\sum_{i\in C}\sum_{p\in P_i^{con}} 2\tau_{ip}x_{ip}^s\Big\}$ is a lower bound on z.

Proof: Makespan (z) is bounded by the maximum of the finishing times of all TVs plus the loading time (β) of the last container. Then, we have to show that $\sum_{i\in C}\sum_{p\in P_i^{con}} 2\tau_{ip}x_{ip}^s$ is a lower bound on the maximum finishing time of each TV. The finishing time of TV s is at least the summation of all transport times for containers that it will load by that TV. □

Proposition 3: $\max_{q\in Q}\Big\{\beta|P_q|\Big\}$ is a lower bound on z.

Proof: Makespan (z) is bounded by the maximum of the finishing times of all QCs that work on the vessel. The finishing time of one QC is at least loading time of all positions that the QC is assigned to ($\beta|P_q|$). So that, the maximum of total loading times is a lower bound on z.□

We now suggest a mathematical model to obtain the lower bound on the FSLP using above propositions. The model uses the same notation and variables of the FSLP model. We introduce a new integer variable, TTS_s, which presents the lower bound on the finishing time of operations for TV s. Now, let us introduce the new model which is called lower bound model, and it is abbreviated as LB-FSLP:

$$\min \alpha\sum_{i\in C}\sum_{p\in P_i^{con}}\sum_{s\in S_p^{pos}} 2\tau_{ip}x_{ip}^s + \gamma\Delta EFT \tag{22}$$

subject to

$$\sum_{p\in P_i^{con}}\sum_{s\in S_p^{pos}} x_{ip}^s = 1 \qquad \forall i \in C \tag{23}$$

$$\sum_{i\in C_p}\sum_{s\in S_p^{pos}} x_{ip}^s = 1 \qquad \forall p \in P \tag{24}$$

$$x_{ip}^s = 0 \qquad \forall i \in C, \forall p \in P, \forall s \in S \setminus S_p^{pos} \tag{25}$$

$$TTS_s = \beta + \sum_{i\in C}\sum_{p\in P_i^{con}} 2\tau_{ip}x_{ip}^s \quad \forall s \in S \tag{26}$$

$$z \geq TTS_s \qquad \forall s \in S \tag{27}$$

$$z \geq \beta|P_q| \qquad \forall q \in Q \tag{28}$$

$$z \geq \left\lceil \frac{\delta_{min}^q}{|S_q|} \right\rceil \qquad \forall q \in Q \tag{29}$$

$$\Delta EFT \geq z - EFT \tag{30}$$

$$TTS_s, z, \Delta EFT \in \{0, .., H-1\} \qquad \forall s \in S \tag{31}$$

$$x_{ip}^s \in \{0, 1\} \qquad \forall i \in C, \forall s \in S, \forall p \in P \tag{32}$$

The optimal solution to (22)-(32) is a lower bound on the FSLP. The objective function (22) is a combination of the cost of TV transportation times and the tardiness cost. Constraints (23)-(25) are interpreted in a similar way as with constraints (2), (3) and (12) of the FSLP model. Constraint (26) sets the lower bound on the finishing time for each TV, constraint (27) uses these variables to obtain a lower bound on the makespan. Constraint (28) sets the lower bound on finishing time for each QC, where $|P_q|$ refers to the number of positions that will be loaded by QC q. Constraint (29) uses the minimum transportation time to obtain the lower bound on the makespan for the vessel. Constraints (31)-(32) define the domains of variables.

6. Heuristic approach

Broadly speaking there are two main decisions to be taken in the FSLP. 1) The service times for the TVs where $End_s - Start_s$ is service time for TV s and 2) the container-assignment for each TV (x_{ip}^s).

The objective function heavily depends on the service times of the TVs, which on the other hand depend on the container-assignment. We propose a Greedy Randomised Adaptive Search Procedure (GRASP) which initially imposes a specific service time to each TV. The scheduling and container-assignment are then created attempting to respect the assigned service times. Within each GRASP iteration, the generated solution is evaluated. Should a solution be promising enough, it is improved using a local search method.

6.1. Construction heuristic

Following the idea of the two main decisions, the heuristic starts by assigning service times for the TVs, and hereafter calculates a container-assignment while trying to adhere to the assigned service times. Given an arbitrary assigned service time for each of the TV $s \in S_q$ of QC $q \in Q$ (defined by $\overline{Start}_s$ and $\overline{End}_s$), a solution can be constructed with the following four steps:

Step 0: Select QC
The construction heuristic builds the solution by considering the QCs in sequential order. The first step is thus to select the next QC q to build the solution for.

Step 1: Assign containers to positions
Let $\langle c, p\rangle$ be the assignment of container $c \in C_p$ to position $p \in P_q$, and $\Upsilon(q)$ be the set of all container-assignments for the positions serviced by QC $q \in Q$. Starting from the first position to be loaded until the last, the assignment of a container is done greedily by selecting the closest available container. Thus, for an arbitrary position p we select container

$$c = \underset{c' \in C_p(x)}{\arg\min} \left(2\tau_{c'p}\right), \tag{33}$$

where $C_p(x)$ is the set of compatible containers for position p that have not yet been assigned in solution x.

Step 2: Assigning TVs to positions
Let $\langle c^*, p^*\rangle \in \Upsilon(q)$ describe the next container-assignment to which a TV needs to be dispatched (initially defined as the container-assignment for the first position in the loading order of QC q).

For the position p^* let $\Upsilon(q, p^*)$ be the set of the next ι container-assignments including $\langle c^*, p^*\rangle$. Here ι is a parameter controlling the size of the set $\Upsilon(q, p^*)$. Also, let S^A be the set of *active* TVs $s \in S_q$. A TV is said to be active if it can service the container-assignment $\langle c^*, p^*\rangle$ without exceeding the assigned end time ($\overline{End}_s$). More formally a TV is active if $\max(a_s + 2\tau_{c^*p^*}, d_{p-1} + \beta) \leq \overline{End}_s$, where a_s is the time TV s is available (i.e. the time at which it finished its last container delivery, or if no deliveries have been assigned $\overline{Start}_s$), and where d_{p-1} is the time at which the QC began to service the previous position. If the QC has not served any position then $d_{p-1} = -\beta$. Should $S^A = \emptyset$ then we will consider $S^A = S_q$.

We now select, through complete enumeration, the sequence of active TVs to service each container-assignment in $\Upsilon(q, p^*)$ which results in the minimum *completion time*. In this context, the completion time is the time in which QC q has finished loading the last container-assignment of $\Upsilon(q, p^*)$. For few TVs and small ι partitions, the exponential growth of the number of sequence combinations is not an issue for the complete enumeration. We only consider the active TVs to prioritise the scheduling of TVs that do not exceed the EFT. This is due to the way TV service times are generated (more details are provided in Section 6.2.2).

Step 3: Assign TV and update times
Given the TV sequence found in Step 2, we only commit to the solution the TV scheduled for container assignment $\langle c^*, p^*\rangle$. Practically we are only assigning the TV to the very next position to load (p^*). Hereafter, the selected TV available time and the QC time are updated (a_s and d_p). The container-assignment $\langle c^*, p^*\rangle$ is then removed from $\Upsilon(q)$. Should $\Upsilon(q) = \emptyset$, go to Step 0 and process the next QC, otherwise go to Step 2.

6.2. GRASP

GRASP (Feo and Resende, 1989, 1995), is a multi-start iterative metaheuristic that combines a constructive phase with an improvement phase. In each iteration, a solution is built from scratch using a randomised construction heuristic, where a random strategy is used to provide diversity. The improvement phase consists of a local search method used to improve the solution found in the constructive phase. The GRASP method has successfully been applied to many optimisation problems, such as the container stowage slot planning problem (Parreno et al., 2016), Vehicle routing problems (Kontoravdis and Bard, 1995) and the quadratic assignment problem (Pardalos and Resende, 1994), among others. A general outline of the GRASP algorithm for the FSLP is presented in Algorithm 1.

The algorithm begins by generating a solution using a randomised version of the construction heuristic described earlier (line 3). The randomised construction heuristic is described in Section 6.2.1.

After a feasible solution has been found, it is then passed to the improvement phase. Two improvement heuristics have been implemented, each focusing on different aspects of the solution and thus complementing each other. The first method is aimed at improving the TVs' schedule, and it is used at every iteration (line 4). The second focuses on the container assignment. However, due to its computational complexity, it is only used when the cost of the candidate solution ($f(x)$) is within κ percentage of the best-found solution cost ($f(x^b)$). The improvement methods are described in details in Section 6.2.3 and Section 6.2.4 respectively.

Algorithm 1 GRASP

1: $x^b \leftarrow \emptyset$
2: **while not** Terminate() **do**
3: $x \leftarrow$ RandomisedConstructionHeuristic()
4: $x \leftarrow$ VehicleReassignment(x)
5: **if** $f(x) < f(x^b)(1+\kappa)$ **then**
6: $x \leftarrow$ ContainerSwap(x)
7: **end if**
8: UpdateBest(x^b, x)
9: **end while**
10: **return** x^b

In each iteration, a new solution is made from scratch, and if the newly generated solution is better than the previous best, it is kept (line 8). Once the termination criterion is reached, the algorithm returns the best-found solution. In our approach, we use the number of iterations as the termination criterion. Following we describe the implementation of each GRASP component in detail.

6.2.1. Randomised construction heuristic

Randomization is included into the construction heuristic (Section 6.1) in three places: the order in which QCs are processed, the container to position assignment (Step 1), and the generation of the service times, which the heuristic is based on.

Randomizing the order in which the QCs are processed (Step 0), results in variations on the assignment of containers to position. For the actual container to position assignment, let ρ_c be a random number in the interval $[0.5; 1.5]$. At each iteration, a random number ρ_c is sampled for every container c. The container c that minimises the driving distance times ρ_c is assigned to the position p, thus effectively changing eq. (33) to (34).

$$c = \underset{c' \in C_p(x)}{\arg\min} \left(\rho_{c'} 2\tau_{c'p}\right) \qquad \forall p \in P_q \tag{34}$$

The last source of randomisation, the generation of service times, also takes care of the adaptive part of the procedure (as explained in Section 6.2.2). Service times are generated on a per QC basis using the following concept. Consider a QC $q \in Q$. We define the total service time to be $\nu_q = \sum_{s \in S_q} \overline{End}_s - \overline{Start}_s$. Since the TVs are operating in parallel, the largest total service time that a QC can have, without any delays, is $|S_q|EFT$. This corresponds to all TVs starting at time 0 and ending at time EFT. The value of ν_q can be used to guide the generation of the service time of each TV. We do so by imposing $\overline{Start}_s = 0$ for all TVs $s \in S_q$. We then assign $\overline{End}_s = EFT$ for as many TVs as possible. Hence exactly $\left\lfloor \frac{\nu_q}{EFT} \right\rfloor$ TVs will have this assignment for QC q. Any residual value $\nu_q - \left\lfloor \frac{\nu_q}{EFT} \right\rfloor$ will be assigned to one TV, while all remaining TVs are assigned $\overline{End}_s = 0$. The service times assignment is the base of the construction heuristic. Consider now Step 2 of the heuristic. The procedure starts by scheduling only the active TVs, i.e. the TVs $s \in S_q$ that can finish the next scheduled container-assignment ($\langle c^*, p^* \rangle$) within $\overline{End}_s$. By doing so, we make sure that no TV is scheduled after the $\overline{End}_s$ unnecessarily, which also conforms to the way the service times are generated. Notice, however, that should this not be possible the heuristic will reset the set of active TVs to be equal to the complete set of TVs (S_q) thus allowing scheduling beyond $\overline{End}_s$.

The randomisation of the service time generation is rooted in the selection of the total service time of each QC (ν_q). At each iteration, the value of ν_q is selected at random within the range $[EFT; |S_q|EFT]$. To better guide the selection of ν_q we partition the range into intervals of size ε, effectively generating the following set of intervals, I_q:

$$\{[EFT; EFT+\varepsilon], [EFT+\varepsilon; EFT+2\varepsilon], \ldots, [|S_q|EFT-\varepsilon; |S_q|EFT]\} \cup$$
$$\{[EFT; EFT], [|S_q|EFT; |S_q|EFT]\}$$

The values of ν_q are then selected at random within one of these range intervals. When the expected finishing time is low, it would most likely be best to assign all available TVs. On the other hand, when EFT is high, one TV operating is likely to be cost optimal. For these reasons, we have also included the ranges $[EFT; EFT]$ and $[|S_q|EFT; |S_q|EFT]$ in the set. We call these range intervals Service Time Intervals (STIs).

6.2.2. Adaptivity

When selecting the value ν_q, a STI (i) is first chosen, and ν_q is selected at uniform within the range interval described by i. The adaptivity of the GRASP method comes from how the probability of choosing a STI adaptively changes throughout the execution of the algorithm. For each STI, let $\mathbb{P}_{iq}$ be the probability of choosing STI i for QC q, calculated using the

roulette wheel selection principle

$$\mathbb{P}_{iq} = \frac{w_i}{\sum_{j \in I_q} w_j} \qquad \forall q \in Q, i \in I_q \tag{35}$$

where w_i are the weights. This probability is adaptively adjusted similarly to the ALNS method described in Ropke and Pisinger (2006).

Throughout the algorithm, we keep track of the best-found solution and its value $\widehat{z}$. The positions to be loaded by QC q are pre-determined, thus we also keep track of the best partial solution scheduling container-assignments for QC q. The cost of a partial QC solution is calculated as

$$\widehat{z_q} = \alpha \sum_{s \in S_q} (End_s - Start_s) + \gamma \max\left(\left(\max_{s \in S_q} End_s\right) + \beta - EFT, 0\right) \tag{36}$$

where $\max\left(\left(\max_{s \in S_q} End_s\right) + \beta - EFT, 0\right)$ is the tardiness of the operation for QC q.

The execution of the algorithm is divided into a number of *segments* i.e. a number of η consecutive iterations. The score obtained by STI i in segment j (denoted π_{ij}) is updated according to the following three parameters; σ_1, σ_2 and σ_3. If a new best solution is found, σ_1 is added to the score for all the chosen STIs that contributed to finding this solution. If the cost for QC q (z_q) is better than the previously best solution for that QC, $\widehat{z_q}$, σ_2 is added to the score for the STI chosen for QC q. Last, if for a QC q the cost (z_q) is within δ percentage of the best for QC q ($\widehat{z_q}$) then σ_3 is added to the score for the chosen STI for QC q.

After η iterations are executed the weights are updated as follows.

$$w_{ij+1} = w_{ij}(1 - r) + r\frac{\pi_{ij}}{\theta_{ij}} \qquad \forall q \in Q, i \in I_q \tag{37}$$

Here w_{ij} is the weight for STI i in segment j, θ_{ij} is the number of times STI i was chosen in segment j and r is the *reaction factor*. The reaction factor controls how quickly the weight adjustment reacts to changes in the effectiveness of each STI.

The adaptivity as described will make the STIs which contributes to finding good solutions more probable. Furthermore, it makes the overall heuristic robust towards different characteristics in the problem that have an impact on the time needed to load the containers for a QC.

6.2.3. Vehicle reassignment

The first improvement method looks at the TV scheduling for a QC q. The TV scheduling described in Section 6.1 Step 3 is myopic at best, and only aims at minimising tardiness. It does not consider TV waiting time and the minimisation of the total service time.

The TV reassignment procedure is aimed at reducing the service times of each TV. Since the service time for TV s is dictated by its first and last container-assignment, we will only consider the possibility of re-assigning these container-assignments to another TV. For a given QC q we generate all the possible re-assignments of containers to TVs (which is at most $2|S_q| - 1$ since we only consider two containers per TV). The re-assignment that best improves the objective is then applied. The procedure restarts every time a new improving re-assignment is found until no new re-assignment is available and all the QCs are processed.

6.2.4. Container swapping

The second improvement method is a local search based on a swap neighbourhood operator. The local search aims at finding improvements in the container assignments. The neighbourhood is defined by all the possible container swaps within the same container class. Here a swap means that two positions exchange containers, and consequently the two TVs scheduled to the positions will change the container they pickup. The neighbourhood operator, however, only evaluates a limited number of swaps to reduce the number of evaluations. The most improving swap is then applied to the solution. The container swapping procedure is further described in Algorithm 2.

Given an input solution x, Algorithm 2 starts by initializing z^* and $Move$ which will hold the best evaluation and swapping move respectively (lines 3-4). The algorithm then proceeds to evaluate the possible swaps. For each class u, in the set of container classes U, we select $|C_u|$ random swaps, where C_u is the set of all containers of class u (lines 5-8). Each swap is evaluated (line 9) and, if it is improving, its value and move are stored in z^* and $Move$ (lines 10-13). The best improving swap is then selected and applied to the solution (lines 17-18). Should we not be able to find such a swap, the procedure terminates and returns the locally improved solution, (lines 2, 19-23)

Calculating a swap improvement in `EvaluateSwapImprovement`(x, c_1, c_2) is the most computationally expensive part of this method. In Appendix A, we provide details of how caching techniques can be used to implement this operation efficiently.

Algorithm 2 `ContainerSwap`

Input: x
Terminate $\leftarrow$ **false**
while not Terminate **do**
  $z^* \leftarrow 0$
  $Move \leftarrow \emptyset$
  for all $u \in U$ **do**
    for $i = 1$ **to** $|C_u|$ **do**
      $c_1 \leftarrow$ Random container $c \in C_u$
      $c_2 \leftarrow$ Random container $c \in C_u \setminus \{c_1\}$
      $z \leftarrow$ `EvaluateSwapImprovement`(x, c_1, c_2)
      if $z > z^*$ **then**
        $z^* \leftarrow z$
        $Move \leftarrow \{c_1, c_2\}$
      end if
      $i \leftarrow i + 1$
    end for
  end for
  if $z^* > 0$ **then**
    $x \leftarrow$ `PerformMove`$(x, Move)$
  else
    Terminate $\leftarrow$ **true**
  end if
end while
return x

7. Computational analysis

We now analyse the performance of each formulation, valid inequalities and the GRASP heuristic. All methods are executed using a 2.30 GHz Intel Xeon E5 Processor and 128GB available memory. Computational times are reported in seconds (s). All models are solved using CPLEX 12.7.0. A time limit of 3 hours is imposed to solve the models with the options of *emphasising optimality*. In default conditions, models are run with four threads.

The GRASP heuristic has been implemented in `Java` 1.8 and has been tuned using the *Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms* (gga) described in Ansótegui et al. (2009). Table 2 describes how we tuned the algorithm. The first two columns describe the parameters, first brief in text then the symbol used. The next two column contains information for the tuning, first which values we tested, then the start value we used. For some of the parameters we tested a discrete set of values, and for others, we tested all integers in a range (symbolised by [min; max]). For the tuner, additional instances have been generated, and the heuristic terminates with the best-found solution after 30 seconds. The value in the last column is the value that the tuner finds perform best.[1]

Table 2: Description of the parameter tuning

Description	Symbol	Test	Start	Tuned
STI length	ε	$\{0.01, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.2\}$EFT	0.1EFT	0.15EFT
Vehicle assignment lookahead	ι	$\{1, 2, 3, 4, 5\}$	4	2
Container swapping use percentage	κ	$\{0, 0.05, 0.1, 0.15, 0.2, \infty\}$	0.1	0.2
Reaction factor	r	$\{0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.2\}$	0.1	0.1
Segment size	η	$\{25, 50, 75, 100, 150, 200, 250, 300, 350, 400, 500\}$	200	25
STI score update: New best	σ_1	$[30; 75]$	50	45
STI score update: New crane best	σ_2	$[20; 50]$	33	36
STI score update: New crane solution within δ of crane best	σ_3	$[10; 45]$	19	17
The δ parameter associated with σ_3	δ	$\{0.01, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.2\}$	0.05	0.2

7.1. Data description

To test the formulations and the heuristic a benchmark set has been generated. The number of containers to be loaded in the instances are either 60, 240, 500 or 1000, corresponding to small, medium and large vessels. To describe different yard structures, we test three different *densities*. The density will affect the travel time to the containers in the yard. The different densities are *Uniform*, *Scattered* and *Less Dense*. For the Uniform case, all containers are stored closely together in the yard, minimising the variance in the travel times. In the scattered case the containers are stored over a larger area, and it implies a higher variance in the travel times. The Less Dense is a mix between these two densities.

[1] The value ∞ for κ means that the container swapping method is used in every iteration

In total 30 instances have been generated. The number of container types ranges from 10 to 100. In the benchmark, there are 3 TVs assigned to a QC, and either 2 or 4 QCs are available depending on the size of the vessel. The QC loading time β is 1 minute in all of the instances. See Tables 3 and 4 for an overview of the instance characteristics.

The expected finishing time (EFT) corresponds to the schedule of the vessel. Vessels with the same number of containers to be loaded will, therefore, have the same EFT, independent of the yard density, and the number of container types.

7.2. Results for the mathematical model and enhancements

In this section, we report the computational results of the FSLP model and analyse the improvements achieved with the enhancements for this formulation. The results cover the lower bounds obtained in the root node of the branch-and-bound tree (x^{Root}), the best lower bound (x^{LP}), and the value of the obtained solution (x^{UB}). The $Gap(x^{UB})$ columns report the relative difference between the solution value and the best lower bound from the model, and lastly $t(s)$ is the computation time in seconds. The first columns in Tables 3 and 4 describe the instance characteristics, with the number of containers ($|C|$), the number of container types ($|CT|$), the number of QCs ($|Q|$) and the density (D). The densities are Less Dense (LD), Scattered (S) and Uniform (U) as described in Section 7.1.

To evaluate the benefit of each of the enhancements, Tables 3 and 4 present the results for different versions of the model. The first one is the version with all of the enhancements added (FSLP+), after which results for the standard model with no enhancements (FSLP) are shown. For the rest of the models, enhancements are removed from the FSLP+ model, e.g. FSLP+ - (19) is the FSLP+ model with constraint (19) removed. Note that enhancements are removed one at a time. This is to show how badly the results are affected when an enhancement is not used in the formulation, and by that see if it is beneficial to add it.

There are many instances for which CPLEX cannot find any feasible solutions within the time limit (10800 s), this is indicated with '-'. Additionally '†' is used to symbolise that the execution was terminated due to lack of memory. In most of these cases no bound or solution could be computed before termination, but in four cases (FSLP+/60/25/2/S, FSLP+/60/10/2/U, FSLP+ - (21)/60/25/2/LD and FSLP+ - (21)/60/25/2/U) a solution had been found, in which case the time reported is the time at which the execution was terminated. To symbolise the best bound, or best solution found for the given instance among the alternative models, the corresponding value is written in **bold**.

Tables 3 and 4 report the model results and the benefits gained by the enhancements of the model, and results clearly show that FSLP+ model outperforms the FSLP model. The FSLP+ model finds stronger bounds, and also better feasible solutions. Looking at the results for all models it can be seen that constraint (21) is the main contributor to the improvement of the bounds, but (17) & (18) also help to improve the bound. The performance of the three models FSLP+, FSLP+ - (19) and FSLP+ - (20) are all comparable to each other. The bounds all coincide with the best found, and they are all found in the root node and not improved hereafter. The value of the solutions are different, but for a given instance the solution values are close to each other for the three models. FSLP+ - (20) seems to find the best solution most often.

For the 500 and 1000 instances, the model becomes intractable to solve; only a few number of feasible solutions are found within 3 hours.

Table 3: Performance of enhancements of the mathematical model. '−' symbolises that no feasible solution was found within the time limit, and '†' is used for the instances where the execution was terminated due to insufficient memory.

				FSLP+					FSLP					FSLP+ - (17)&(18)				
$\|C\|$	$\|CT\|$	$\|Q\|$	D	x^{Root}	x^{LB}	x^{UB}	$Gap(x^{UB})$	$t(s)$	x^{Root}	x^{LB}	x^{UB}	$Gap(x^{UB})$	$t(s)$	x^{Root}	x^{LB}	x^{UB}	$Gap(x^{UB})$	$t(s)$
60	10	2	*LD*	**1500**	**1500**	1775	15.5%	10800	10	194	1735	88.8%	10800	1440	1440	**1710**	15.8%	10800
60	10	2	*S*	**1020**	**1020**	1030	1.0%	10800	30	93	1030	90.9%	10800	**1020**	**1020**	**1020**	0.0%	8246
60	10	2	*U*	**1640**	**1640**	1920	14.6%	9004†	41	139	2250	93.8%	10800	**1640**	**1640**	1875	12.5%	10800
60	25	2	*LD*	**2030**	**2030**	2345	13.4%	10800	0	336	2480	86.5%	10800	1790	1790	2285	21.7%	10800
60	25	2	*S*	**1360**	**1360**	1470	7.5%	6215†	0	257	1905	86.5%	10800	**1360**	**1360**	**1405**	3.2%	10800
60	25	2	*U*	**1490**	**1490**	1520	2.0%	10800	0	267	2225	88.0%	10800	**1490**	**1490**	**1510**	1.3%	10800
			Average				9.0%	9737				89.1%	10800				9.1%	10374
240	20	2	*LD*	**7850**	**7850**	14795	46.9%	10800	0	100	20170	99.5%	10800	6530	6530	-	-	10800
240	20	2	*S*	**4440**	**4440**	8225	46.0%	10800	0	50	109905	100.0%	10800	**4440**	**4440**	9040	50.9%	10800
240	20	2	*U*	**6720**	**6720**	11765	42.9%	10800	0	60	19765	99.7%	10800	**6720**	**6720**	11435	41.2%	10800
240	60	2	*LD*	**8230**	**8230**	11035	25.4%	10800	0	142	16565	99.1%	10800	6850	6850	**10650**	35.7%	10800
240	60	2	*S*	**5280**	**5280**	**6555**	19.5%	10800	0	79	11095	99.3%	10800	**5280**	**5280**	7310	27.8%	10800
240	60	2	*U*	**7250**	**7250**	10845	33.1%	10800	0	179	14005	98.7%	10800	**7250**	**7250**	10630	31.8%	10800
			Average				35.6%	10800				99.4%	10800				37.5%	10800
500	20	4	*LD*	**14390**	**14390**	-	-	10800	0	0	-	-	10800	12110	12110	-	-	10800
500	20	4	*S*	**8250**	**8250**	-	-	10800	0	0	-	-	10800	**8250**	**8250**	-	-	10800
500	20	4	*U*	**14350**	**14350**	-	-	10800	0	0	-	-	10800	13090	13090	-	-	10800
500	60	4	*LD*	**14460**	**14460**	-	-	10800	0	200	-	-	10800	12930	12930	-	-	10800
500	60	4	*S*	**11840**	**11840**	-	-	10800	0	60	-	-	10800	**11840**	**11840**	-	-	10800
500	60	4	*U*	**15500**	**15500**	264170	94.1%	10800	0	233	**30690**	99.2%	10800	14240	14240	-	-	10800
500	100	4	*LD*	**15390**	**15390**	-	-	10800	0	288	40225	99.3%	10800	13860	13860	138790	90.0%	10800
500	100	4	*S*	**9490**	**9490**	-	-	10800	0	140	24140	99.4%	10800	**9490**	**9490**	16895	43.8%	10800
500	100	4	*U*	**16230**	**16230**	**22625**	28.3%	10800	0	240	36700	99.3%	10800	14880	14880	-	-	10800
			Average				61.2%	10800				99.3%	10800				66.9%	10800
1000	20	4	*LD*	†	†	†	†	†	†	†	†	†	†	†	†	†	†	†
1000	20	4	*S*	†	†	†	†	†	†	†	†	†	†	†	†	†	†	†
1000	20	4	*U*	†	†	†	†	†	†	†	†	†	†	†	†	†	†	†
1000	60	4	*LD*	**27070**	**27070**	-	-	10800	0	0	-	-	10800	24460	24460	-	-	10800
1000	60	4	*S*	**17950**	**17950**	-	-	10800	0	0	-	-	10800	**17950**	**17950**	-	-	10800
1000	60	4	*U*	**40230**	**40230**	-	-	10800	0	0	-	-	10800	34320	34320	-	-	10800
1000	100	4	*LD*	**28740**	**28740**	-	-	10800	0	0	-	-	10800	25830	25830	-	-	10800
1000	100	4	*S*	**17840**	**17840**	-	-	10800	0	0	-	-	10800	**17840**	**17840**	-	-	10800
1000	100	4	*U*	**32180**	**32180**	-	-	10800	50	50	-	-	10800	29090	29090	-	-	10800
			Average					10800					10800					10800
Average							27.9%	10564				95.5%	10800				28.9%	10705

Table 4: Performance of enhancements of the mathematical model. '−' symbolises that no feasible solution was found within the time limit, and '†' is used for the instances where the execution was terminated due to insufficient memory.

				FSLP+ - (19)					FSLP+ - (20)					FSLP+ - (21)				
$\|C\|$	$\|CT\|$	$\|Q\|$	D	x^{Root}	x^{LB}	x^{UB}	$Gap(x^{UB})$	$t(s)$	x^{Root}	x^{LB}	x^{UB}	$Gap(x^{UB})$	$t(s)$	x^{Root}	x^{LB}	x^{UB}	$Gap(x^{UB})$	$t(s)$
60	10	2	*LD*	**1500**	**1500**	1800	16.7%	10800	**1500**	**1500**	1720	12.8%	10800	101	222	1750	87.3%	10800
60	10	2	*S*	**1020**	**1020**	1025	0.5%	10800	**1020**	**1020**	**1020**	0.0%	5612	0	105	1030	89.8%	10800
60	10	2	*U*	**1640**	**1640**	**1790**	8.4%	10800	**1640**	**1640**	1820	9.9%	10800	0	121	1955	93.8%	10800
60	25	2	*LD*	**2030**	**2030**	2350	13.6%	10800	**2030**	**2030**	**2275**	10.8%	10800	241	632	2535	75.1%	7615†
60	25	2	*S*	**1360**	**1360**	1440	5.6%	10800	**1360**	**1360**	1580	13.9%	10800	0	307	1760	82.6%	10800
60	25	2	*U*	**1490**	**1490**	1530	2.6%	10800	**1490**	**1490**	1515	1.7%	10800	0	401	1740	76.9%	6568†
			Average				7.9%	10800				8.2%	9935				84.3%	9564
240	20	2	*LD*	**7850**	**7850**	14795	46.9%	10800	**7850**	**7850**	**12700**	38.2%	10800	1320	1403	19600	92.8%	10800
240	20	2	*S*	**4440**	**4440**	8225	46.0%	10800	**4440**	**4440**	**6400**	30.6%	10800	0	30	10045	99.7%	10800
240	20	2	*U*	**6720**	**6720**	-	-	10800	**6720**	**6720**	**11355**	40.8%	10800	0	30	17710	99.8%	10800
240	60	2	*LD*	**8230**	**8230**	10730	23.3%	10800	**8230**	**8230**	11705	29.7%	10800	1380	1500	15675	90.4%	10800
240	60	2	*S*	**5280**	**5280**	6680	21.0%	10800	**5280**	**5280**	6630	20.4%	10800	11	73	8265	99.1%	10800
240	60	2	*U*	**7250**	**7250**	**10185**	28.8%	10800	**7250**	**7250**	13330	45.6%	10800	0	170	16735	99.0%	10800
			Average				33.2%	10800				34.2%	10800				96.8%	10800
500	20	4	*LD*	**14390**	**14390**	-	-	10800	**14390**	**14390**	-	-	10800	2280	2280	-	-	10800
500	20	4	*S*	**8250**	**8250**	-	-	10800	**8250**	**8250**	-	-	10800	0	0	-	-	10800
500	20	4	*U*	**14350**	**14350**	-	-	10800	**14350**	**14350**	-	-	10800	1260	1260	-	-	10800
500	60	4	*LD*	**14460**	**14460**	-	-	10800	**14460**	**14460**	-	-	10800	1530	1750	-	-	10800
500	60	4	*S*	**11840**	**11840**	-	-	10800	**11840**	**11840**	**340870**	96.5%	10800	0	50	-	-	10800
500	60	4	*U*	**15500**	**15500**	-	-	10800	**15500**	**15500**	-	-	10800	1260	1501	-	-	10800
500	100	4	*LD*	**15390**	**15390**	**21730**	29.2%	10800	**15390**	**15390**	22695	32.2%	10800	1530	1753	37720	95.4%	10800
500	100	4	*S*	**9490**	**9490**	**16220**	41.5%	10800	**9490**	**9490**	-	-	10800	0	140	233945	99.9%	10800
500	100	4	*U*	**16230**	**16230**	-	-	10800	**16230**	**16230**	-	-	10800	1350	1629	41490	96.1%	10800
			Average				35.3%	10800				64.4%	10800				97.1%	10800
1000	20	4	*LD*	†	†	†	†	†	†	†	†	†	†	†	†	†	†	†
1000	20	4	*S*	†	†	†	†	†	†	†	†	†	†	†	†	†	†	†
1000	20	4	*U*	†	†	†	†	†	†	†	†	†	†	†	†	†	†	†
1000	60	4	*LD*	-	-	-	-	10800	**27070**	**27070**	-	-	10800	2610	2610	-	-	10800
1000	60	4	*S*	**17950**	**17950**	-	-	10800	**17950**	**17950**	-	-	10800	0	0	-	-	10800
1000	60	4	*U*	**40230**	**40230**	-	-	10800	**40230**	**40230**	-	-	10800	5910	5910	-	-	10800
1000	100	4	*LD*	**28740**	**28740**	-	-	10800	**28740**	**28740**	-	-	10800	2910	2910	-	-	10800
1000	100	4	*S*	**17840**	**17840**	-	-	10800	**17840**	**17840**	-	-	10800	0	0	-	-	10800
1000	100	4	*U*	**32180**	**32180**	-	-	10800	**32180**	**32180**	-	-	10800	3090	3090	-	-	10800
			Average					10800					10800					10800
Average							21.8%	10800				27.4%	10608				91.9%	10525

7.3. Heuristic results

We evaluate the performance of GRASP heuristic and compare it with the results from the formulation FSLP+ and the lower bound model (LB-FSLP) presented in Section 5. Table 5 reports the results of the FSLP+ formulation, the lower bounds from LB-FSLP and the GRASP results on the instances from the benchmark. In Table 5, x^R is the lower bound from the LB-FSLP model and $t(s)$ reports the computational time to obtain these lower bounds. For the FSLP+ model, x^{LB} and x^{UB} is the lower bound, and upper bound obtained. $Gap(x^{UB})$ is the gap, comparing the best feasible solution with the best found lower bound. For the heuristic, each instance is solved ten times, to account for the randomness. The best solution and average solution for the ten runs are reported. For the GRASP heuristic, the following are reported in Table 5; the best solution (x^b), average solution ($\bar{x}$), best and average gap with respect to LB-FSLP ($Gap(x^b)$, $Gap(\bar{x})$) and the run time in seconds ($\bar{t}$). For the instances where the average solution ($\bar{x}$) is better than the FSLP+ formulation upper bound (x^{UB}), the average solution is written in **bold**.

We first evaluate the lower bounds obtained with LB-FSLP model. Table 5 points out that the lower bounds obtained with the LB-FSLP model are no worse than the x^{LB} values from the FSLP+ model, in some cases the LB-FSLP model even finds a better lower bound. Moreover, the bounds are computed in just 3 seconds on average. This indicates that the lower bounds obtained with LB-FSLP can be used to evaluate the performance of the GRASP heuristic.

Table 5: Results overview: Comparison between the formulations and the GRASP heuristic. '−' is used when no feasible solution was found within the timelimit

				LB-FSLP		FSLP+			GRASP				
$\|C\|$	$\|CT\|$	$\|Q\|$	D	x^R	$t(s)$	x^{LB}	x^{UB}	$Gap(x^{UB})$	x^b	$\bar{x}$	$Gap(x^b)$	$Gap(\bar{x})$	$\bar{t}$ (s)
60	10	2	*LD*	1530	0.1	1500	1775	15.49%	1805	1814.5	15.24%	15.68%	9.9
60	10	2	*S*	1020	0.0	1020	1030	0.97%	1060	1067	3.77%	4.40%	7.5
60	10	2	*U*	1670	0.1	1640	1920	14.58%	1895	**1902.5**	11.87%	12.22%	8.8
60	25	2	*LD*	2060	0.0	2030	2345	13.43%	2435	2435	15.40%	15.40%	10.4
60	25	2	*S*	1360	0.0	1360	1470	7.48%	1425	**1436.5**	4.56%	5.32%	10.1
60	25	2	*U*	1490	0.0	1490	1520	1.97%	1545	1552.5	3.56%	4.02%	8.5
			Average		**0.1**						**9.07%**	**9.51%**	**9.2**
240	20	2	*LD*	7880	0.3	7850	14795	46.94%	9430	**9448**	16.44%	16.60%	80.5
240	20	2	*S*	4440	0.1	4440	8225	46.02%	4790	**4807.5**	7.31%	7.64%	36.2
240	20	2	*U*	6720	0.7	6720	11765	42.88%	8140	**8333.5**	17.44%	19.35%	36.8
240	60	2	*LD*	8260	0.1	8230	11035	25.42%	10105	**10123**	18.26%	18.40%	131.1
240	60	2	*S*	5280	0.1	5280	6555	19.45%	5660	**5706.5**	6.71%	7.47%	53.4
240	60	2	*U*	7580	0.2	7250	10845	33.15%	9065	**9140.5**	16.38%	17.07%	52.8
			Average		**0.2**						**13.76%**	**14.42%**	**65.1**
500	20	4	*LD*	14420	1.4	14390	-	-	15585	**15639**	7.48%	7.79%	538.7
500	20	4	*S*	8250	1.3	8250	-	-	9020	**9129.5**	8.54%	9.63%	75.0
500	20	4	*U*	14380	2.0	14350	-	-	15585	**15594.5**	7.73%	7.79%	381.9
500	60	4	*LD*	14520	4.2	14460	-	-	16130	**16225.5**	9.98%	10.51%	648.3
500	60	4	*S*	11840	0.2	11840	-	-	13005	**13080**	8.96%	9.48%	72.2
500	60	4	*U*	15530	0.2	15500	264170	94.13%	17125	**17173**	9.31%	9.57%	287.1
500	100	4	*LD*	15450	0.5	15390	-	-	17475	**17562**	11.59%	12.03%	627.7
500	100	4	*S*	9490	0.1	9490	-	-	10425	**10547.5**	8.97%	10.02%	73.6
500	100	4	*U*	16290	0.6	16230	22625	28.27%	18495	**18544**	11.92%	12.15%	282.8
			Average		**1.2**						**9.39%**	**9.89%**	**331.9**
1000	20	4	*LD*	21990	9.4	-	-	-	24225	**24277**	9.23%	9.42%	1984.0
1000	20	4	*S*	14570	1.7	-	-	-	16230	**16414**	10.23%	11.23%	292.2
1000	20	4	*U*	25980	49.9	-	-	-	28295	**28354**.5	8.18%	8.37%	1730.9
1000	60	4	*LD*	27100	3.2	27070	-	-	30000	**30074**	9.67%	9.89%	2821.9
1000	60	4	*S*	17950	0.7	17950	-	-	20305	**20599**	11.60%	12.85%	174.8
1000	60	4	*U*	40260	2.1	40230	-	-	44055	**44095.5**	8.61%	8.70%	1209.3
1000	100	4	*LD*	28770	1.7	28740	-	-	32330	**32372**	11.01%	11.13%	3129.7
1000	100	4	*S*	17840	0.5	17840	-	-	20150	**20347**	11.46%	12.32%	177.2
1000	100	4	*U*	32210	1.4	32180	-	-	36160	**36228**	10.92%	11.09%	2194.8
			Average		**7.8**						**10.10%**	**10.56%**	**1523.9**
Average					**2.8**						**10.41%**	**10.92%**	**571.6**

The results in Table 5 show the GRASP heuristic finds feasible solutions for all of the instances, with an average gap of 10.9% in approx. 10 minutes on average. The gap calculates the relative difference to the lower bound, as seen in Section 7.2 only one instance has been solved to optimality, for the rest of the instances we have no indication of the quality of this lower bound, and how far it is from being optimal. The quality of the solutions found by the GRASP heuristic is stable with respect to the number of containers.

Looking at the 500 and 1000 container instances, we see a clear tendency; the *Less Dense* instances require considerably more effort to solve, compared with the *Scattered* instances. The effort needed for the *Uniform* instances lies in between the two others. By looking at the underlying data, the explanation can be found in the time spent on the Improvement Phase. Excluding that time, there are only small deviations in the total time used.

Figure 3 shows how much the two improvement methods contribute to the quality of the final solution. The plots show how the average gap converges over time when using both improvement methods, only the container swapping method, only the vehicle reassignment method and no improvement methods. For each setting, the data is grounded on ten runs of each instance. The plots show that of the two, the container swapping method improves the solution the most, but is also the most time consuming one. From the plot, it is also clear that both of the improvement methods improves the solution noticeably.

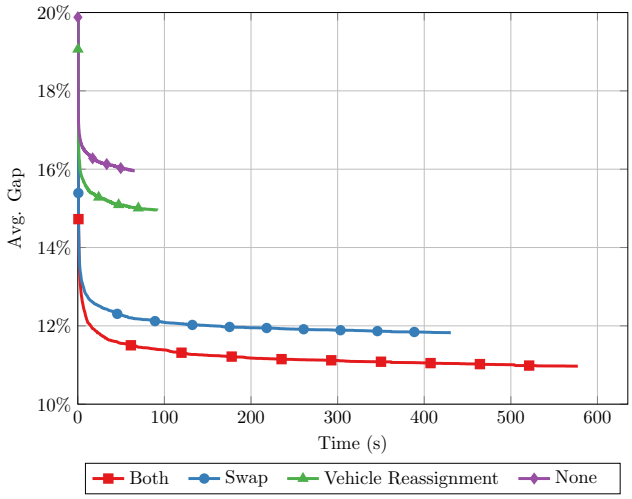


Figure 3: The impact of the improvement methods on time, and solution quality

To visualise the impact of the STIs, Figure 4 shows the probabilities for QC 2 during a single execution of the algorithm on the instance $60/25/2/LD$. The figure shows the probability of selecting a specific STI for the QC in an iteration. The probabilities are shown for 3 of the STIs, for the remaining STIs only a few improvements are found, and thus they become less and less probable. The probabilities for the STIs not shown in Figure 4 are below 0.1% from iteration 5000 until the end. The legends describe the range of the STI, i.e. the interval in which ν_q is randomly sampled when the STI is selected. Figure 4 shows that the most probable STIs (for this QC/instance combination) are towards the end of the full range for ν_q. The interval range for the high probability STIs is close to each other. This is as expected; if one STI provides good results, the following, or preceding is likely to perform semi-good as well.

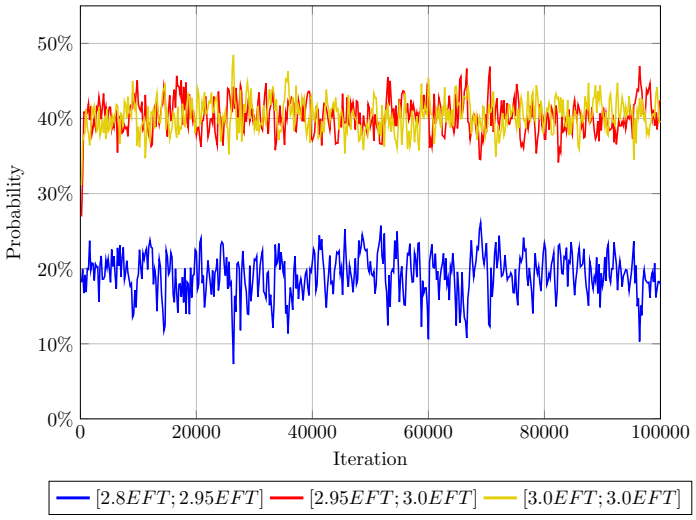
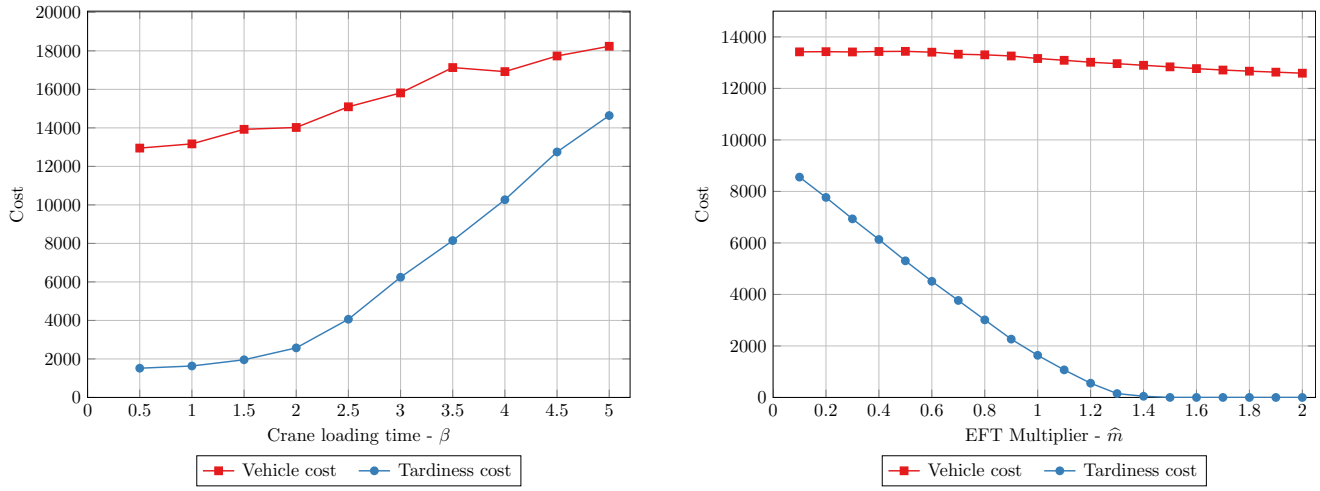


Figure 4: The STI probability during a single execution of the algorithm on the instance $60/25/2/LD$ for QC 2

There are 2 parts to the objective function (1), first a cost of TVs services ($\alpha \sum_{s \in S}(End_s - Start_s)$) and a tardiness cost ($\gamma \Delta EFT$). Figure 5 shows the impact on the cost structure when changing the QC loading time, β and the EFT.

Figure 5a analyses the impact of changing β and Figure 5b of changing the EFT. The data is an average of 10 runs of the heuristic on every instance. Figure 5a shows what we would expect, increasing β increases both the TV service cost and the tardiness cost. This is important for the terminal as crane operators will load the containers faster/slower depending on their skill level. Figure 5b shows the impact of changing the EFT. Here the EFT is recalculated as $\widehat{EFT} = \widehat{m}EFT$, where $\widehat{m}$ is a multiplier and EFT is the original EFT. Intuitively you would expect that a lower EFT means a higher tardiness cost, which Figure 5b confirms. Increasing the EFT mostly has an impact on the TV service cost for an instance when the tardiness cost is 0 for that instance. This makes sense as fewer TVs can be used, and the loading be completed as expected. Using fewer TVs means less unproductive waiting time incurred by the loading order and β.



(a) Impact of β on the cost

(b) Impact of the EFT on the cost

Figure 5: Impact of β and the EFT on the cost structure.

7.4. Hierarchical vs integrated planning: value of integration

We also investigate the cost savings with the integration of the operational stowage planning and TV assignment/scheduling. We compare results of the integrated FSLP with the hierarchical planning method. The hierarchical planning is simulated in two stages. In the first stage, we solve the lower bound model (LB-FSLP in Section 5) where the assignment decisions are solely made without considering the feasible TV scheduling.

The assignment decisions made in (22)-(32) will always generate feasible solutions for the integrated FSLP. This is because the remainder problem is a TV scheduling problem with predetermined assignments (x^s_{ip}). In the second stage, we fix the assignment variables obtained in the first stage and solve the TV scheduling problem. With the assignment fixed, the TV scheduling problem can be solved as a Linear Program. Define $t_p \in \mathbb{R}^+$ as the time the container for position p is dropped in front of the QC. Let c_p be the container to be loaded into position $p \in P$ (which is known). Let f_s and l_s be the first and last position served by TV $s \in S$ and let $\mathrm{A}(s)$ describe the full container/position assignment for that TV. With this, the TV scheduling problem can be modelled as follows:

$$\texttt{Min Z} = \alpha \sum_{s \in S} t_{l_s} - (t_{f_s} - 2\tau_{c_{f_s},f_s}) + \gamma \Delta EFT \tag{38}$$

subject to

$$t_p \geq t_{\bar{p}} + \beta \qquad \forall q \in Q, p \in P_q \setminus \{1\}, \bar{p} \prec\prec p \tag{39}$$

$$t_p \geq 2\tau_{c(p),p} \qquad \forall s \in S, p = f_s \tag{40}$$

$$t_p \geq t_{Prev(s,p)} + 2\tau_{c_p,p} \qquad \forall s \in S, p \in \mathrm{A}(s) \setminus \{f_s\} \tag{41}$$

$$z \geq t_p + \beta \qquad \forall q \in Q, p = |P_q| \tag{42}$$

$$\Delta EFT \geq z - EFT \tag{43}$$

The objective function (38) can be read in the same way as for the FSLP model. Here t_{l_s} is equivalent to End_s, and $(t_{f_s} - 2\tau_{c_{f_s},f_s})$ is equivalent to $Start_s$. In constraint (39), $\bar{p}$ is the position loaded just before position p, and the constraint thus ensures that the loading time is respected. Constraints (40) and (41) ensure the transportation times are respected. For all vehicles, constraint (40) makes sure that the transportation time to the first assignment is respected. For all other assignments, constraint (41) applies. Constraint (41) ensures that the earliest a TV can drop a container in front of the

QC is the time it dropped its last container ($t_{Prev(s,p)}$) plus the time it takes to get the current container. Constraints (42)-(43) are equivalent to constraints (11) and (13) of the FSLP model.

Table 6: Impact analysis of integrated planning for the FSLP

				Hierarchical				Integrated		Value of Integration	
$\|C\|$	$\|CT\|$	$\|Q\|$	D	x^{UB}	$t_1(s)$	$t_2(s)$	$t(s)$	x^{UB}	$\bar{t}(s)$	Δ	%
60	10	2	*LD*	3585	0.1	0.0	0.1	1814.5	9.9	1770.5	49.39%
60	10	2	*S*	2010	0.0	0.0	0.0	1067.0	7.5	943.0	46.92%
60	10	2	*U*	4105	0.1	0.0	0.1	1902.5	8.8	2202.5	53.65%
60	25	2	*LD*	3895	0.0	0.0	0.0	2435.0	10.4	1460.0	37.48%
60	25	2	*S*	3530	0.0	0.0	0.0	1436.5	10.1	2093.5	59.31%
60	25	2	*U*	3760	0.0	0.0	0.0	1552.5	8.5	2207.5	58.71%
		Average									**50.91%**
240	20	2	*LD*	20880	0.3	0.0	0.3	9448.0	80.5	11432.0	54.75%
240	20	2	*S*	10975	0.1	0.0	0.1	4807.5	36.2	6167.5	56.20%
240	20	2	*U*	17455	0.7	0.0	0.7	8333.5	36.8	9121.5	52.26%
240	60	2	*LD*	19530	0.1	0.0	0.1	10123.0	131.1	9407.0	48.17%
240	60	2	*S*	15245	0.1	0.0	0.1	5706.5	53.4	9538.5	62.57%
240	60	2	*U*	20215	0.2	0.0	0.2	9140.5	52.8	11074.5	54.78%
		Average									**54.79%**
500	20	4	*LD*	33755	1.4	0.0	1.4	15639.0	538.7	18116.0	53.67%
500	20	4	*S*	19320	1.3	0.0	1.3	9129.5	75.0	10190.5	52.75%
500	20	4	*U*	32180	2.0	0.0	2.0	15594.5	381.9	16585.5	51.54%
500	60	4	*LD*	30235	4.2	0.0	4.2	16225.5	648.3	14009.5	46.34%
500	60	4	*S*	26185	0.2	0.0	0.2	13080.0	72.2	13105.0	50.05%
500	60	4	*U*	32395	0.2	0.0	0.2	17173.0	287.1	15222.0	46.99%
500	100	4	*LD*	32590	0.5	0.0	0.5	17562.0	627.7	15028.0	46.11%
500	100	4	*S*	21270	0.1	0.0	0.1	10547.5	73.6	10722.5	50.41%
500	100	4	*U*	36815	0.6	0.0	0.6	18544.0	282.8	18271.0	49.63%
		Average									**49.72%**
1000	20	4	*LD*	55815	9.4	0.0	9.4	24277.0	1984.0	31538.0	56.50%
1000	20	4	*S*	33375	1.7	0.0	1.7	16414.0	292.2	16961.0	50.82%
1000	20	4	*U*	58910	49.9	0.0	49.9	28354.5	1730.9	30555.5	51.87%
1000	60	4	*LD*	63880	3.2	0.0	3.2	30074.0	2821.9	33806.0	52.92%
1000	60	4	*S*	44300	0.7	0.0	0.7	20599.0	174.8	23701.0	53.50%
1000	60	4	*U*	82770	2.1	0.0	2.1	44095.5	1209.3	38674.5	46.73%
1000	100	4	*LD*	64735	1.7	0.0	1.7	32372.0	3129.7	32363.0	49.99%
1000	100	4	*S*	43030	0.5	0.0	0.5	20347.0	177.2	22683.0	52.71%
1000	100	4	*U*	69275	1.4	0.0	1.4	36228.0	2194.8	33047.0	47.70%
		Average									**51.42%**
Average											**51.48%**

In Table 6, instance properties are reported in the first four columns. The table is divided into sections presenting hierarchical, integrated and value of integration results. Column x^{UB} presents objective function values, while t variants present the time to obtain the x^{UB} values. In hierarchical planning, $t(s)$ is the sum of two values, $t_1(s)$ and $t_2(s)$. The first value ($t_1(s)$) is the time to run first stage model (i.e. LB-FSLP), while the second value ($t_2(s)$) is the running time of the second stage model ((38) - (43)). Column Δ points out the cost reduction achieved (cost savings) by solving the integrated problem, while % is the percentage of decrease in the cost value (i.e. (Hierarchical-Integrated)/Hierarchical).

Results show that the average cost savings through integration are 51.48%. This suggests that there is a significant potential for savings for the terminal operators with such an integrated problem. Instances with 1000 containers obtain 51.42% savings, while instances with 500, 240 and 60 containers result in 49.72%, 54.79% and 50.91% savings, respectively.

8. Conclusion and future research direction

In this paper, a novel integrated container terminal problem is proposed. This problem focuses on the ship loading operations and aims at integrating the aspects of operational stowage planning with assignment and scheduling of transport vehicles. The problem has been formulated as a mathematical model. To improve the model, novel enhancements are described, and the computational results show that they improve the performance of the mathematical model. However, the exact method becomes computationally intractable for real-life instances. To deal with this, a GRASP heuristic has been implemented. The GRASP heuristic is shown to be scalable and can be used to find quality solutions in reasonable time.

The benefit of solving the integrated problem rather than in a hierarchical fashion has also been investigated. Results show that significant cost savings can be achieved with an efficient solution to the integrated problem.

There are many strong future research directions for both the problem definition and the solution method. With respect to the problem, it would be interesting to relax some of the assumptions in the current model. One promising research path is to integrate the optimisation of the load sequencing within the FSLP. This extension will make the problem more complicated, and the stability of the vessel should be carefully ensured during loading operations. However, the careful implementation of novel solution methods might obtain further cost savings. Another very promising research direction is to allow vehicle pooling and not restrict a given vehicle to only work for a single QC. Such an extension will allow for better utilisation of the TVs. Researchers could consider loading and unloading operations simultaneously with TV pooling where some QCs do loading while others do unloading and share TVs, or double cycling can be allowed to increase the utilisation of the QCs. The FSLP is strongly related to the yard planning problems. An interesting research direction is to integrate ship loading operations with yard allocation decisions considering yard congestion (e.g. Zhen (2016), Zhen et al. (2016b)). A terminal allocation problem with inter-terminal transhipment flows/movements, e.g. Zhen et al. (2016a), can also be attached to these problems. Although a terminal with transfer vehicles is considered in the paper, the problem can be adapted to terminals with different yard transport equipment (e.g. Automated Guided Vehicles (AGVs), Automated Lifting Vehicles (ALVs), etc.) where the operational planning and energy consumption of different yard equipment can be considered.

In reality, terminal optimisation problems are stochastic. In the FSLP we consider the following parameters as being deterministic, even though they realistically are stochastic: the crane productivity, travelling times (depending on the yard congestion as analysed in Zhen (2016)), and the number of containers to handle (depending on when the plan is devised). Before trying to solve a stochastic variant, we believe it is vital to gain as much knowledge as possible from the deterministic version, but the stochastic problem would certainly be a very interesting research direction.

Regarding the solution method, we see three research perspectives. The first is to improve the exact solution method, either by decomposing the problem or reformulating the compact model. One promising idea is to solve the problem using delayed column generation. For this, there are two possible decomposition methods, one based on *TV-plans* and another one based on *QC-plans*. More work is needed to properly evaluate which version would perform better with regards to the strength of the Linear Programming (LP) relaxation and the difficulty of solving the pricing problem. Secondly, it may be possible to improve the heuristic procedure, and lastly, new methods for calculating lower bounds could be examined.

Acknowledgments

The authors would like to thank the associate editor and three reviewers for insightful comments and suggestions. This project was supported by the Danish Maritime Cluster, the Innovation Fond Denmark (1313-00005B-GREENSHIP) and the A/S Dampskibsselskabet Orients Fond (SmartLoad project). The first author was also supported by M4061473 at Nanyang Technological University.

Supplementary material

Supplementary material associated with this article can be found in the online version.

References

Alvarez, J.F., 2006. A heuristic for vessel planning in a reach stacker terminal. Journal of Maritime Research 3, 3–16.

Alvarez, J.F., 2008. Optimization Algorithms for Maritime Terminal and Fleet Management. Ph.D. thesis. Universitat Pompeu Fabra.

Ambrosino, D., Sciomachen, A., 2003. Impact of yard organisation on the master bay planning problem. Maritime Economics & Logistics 5, 285–300.

Ambrosino, D., Sciomachen, A., Tanfani, E., 2004. Stowing a containership: the master bay plan problem. Transportation Research Part A: Policy and Practice 38, 81 – 99.

Ansótegui, C., Sellmann, M., Tierney, K., 2009. A gender-based genetic algorithm for the automatic configuration of algorithms, in: Gent, I.P. (Ed.), Principles and Practice of Constraint Programming - CP 2009: 15th International Conference, CP 2009 Lisbon, Portugal, September 20-24, 2009 Proceedings. Springer Berlin Heidelberg, pp. 142–157.

Bian, Z., Shao, Q., Jin, Z., 2016. Optimization on the container loading sequence based on hybrid dynamic programming. Transport 31, 440–449.

Bierwirth, C., Meisel, F., 2015. A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. European Journal of Operational Research 244, 675 – 689.

Bish, E., Chen, F., Leong, Y., Nelson, B., Ng, J., Simchi-Levi, D., 2005. Dispatching vehicles in a mega container terminal. OR Spectrum 27, 491–506.

Carlo, H.J., Vis, I.F., Roodbergen, K.J., 2014a. Storage yard operations in container terminals: Literature overview, trends, and research directions. European Journal of Operational Research 235, 412 – 430. Maritime Logistics.

Carlo, H.J., Vis, I.F., Roodbergen, K.J., 2014b. Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. European Journal of Operational Research 236, 1 – 13.

Carlo, H.J., Vis, I.F.A., Roodbergen, K.J., 2013. Seaside operations in container terminals: literature overview, trends, and research directions. Flexible Services and Manufacturing Journal 27, 224–262.

Ding, Y., Wei, X.J., Yang, Y., Gu, T.Y., 2017. Decision support based automatic container sequencing system using heuristic rules. Cluster Computing 20, 239–252.

Feo, T.A., Resende, M.G., 1989. A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8, 67 – 71.

Feo, T.A., Resende, M.G.C., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133.

Goodchild, A., Daganzo, C., 2007. Crane double cycling in container ports: Planning methods and evaluation. Transportation Research Part B: Methodological 41, 875 – 891.

Imai, A., Nishimura, E., Papadimitriou, S., Sasaki, K., 2002. The containership loading problem. International Journal of Maritime Economics 4, 126–148.

Imai, A., Sasaki, K., Nishimura, E., Papadimitriou, S., 2006. Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. European Journal of Operational Research 171, 373–389.

Iris, Ç., Pacino, D., 2015. A survey on the ship loading problem, in: Corman, F., Voss, S., Negenborn, R.R. (Eds.), Computational Logistics. Springer International Publishing. volume 9335 of *Lecture Notes in Computer Science*, pp. 238–251.

Iris, Ç., Pacino, D., Ropke, S., 2017. Improved formulations and an adaptive large neighborhood search heuristic for the integrated berth allocation and quay crane assignment problem. Transportation Research Part E: Logistics and Transportation Review 105, 123 – 147.

Iris, Ç., Pacino, D., Ropke, S., Larsen, A., 2015. Integrated berth allocation and quay crane assignment problem: Set partitioning models and computational results. Transportation Research Part E: Logistics and Transportation Review 81, 75 – 97.

Ji, M., Guo, W., Zhu, H., Yang, Y., 2015. Optimization of loading sequence and rehandling strategy for multi-quay crane operations in container terminals. Transportation Research Part E: Logistics and Transportation Review 80, 1 – 19.

Jiang, X.J., Jin, J.G., 2017. A branch-and-price method for integrated yard crane deployment and container allocation in transshipment yards. Transportation Research Part B: Methodological 98, 62 – 75.

Jung, S., Kim, K., 2006. Load scheduling for multiple quay cranes in port container terminals. Journal of Intelligent Manufacturing 17, 479–492.

Kim, K.H., Kang, J.S., Ryu, K.R., 2004. A beam search algorithm for the load sequencing of outbound containers in port container terminals. OR Spectrum 26, 93–116.

Kim, K.H., Kim, K.Y., 1999. An optimal routing algorithm for a transfer crane in port container terminals. Transportation Science 33, 17–33.

Kim, K.H., Lee, H., 2015. Container Terminal Operation: Current Trends and Future Challenges. Springer International Publishing, Cham. pp. 43 – 73.

Kim, K.H., Park, Y.M., 2004. A crane scheduling method for port container terminals. European Journal of Operational Research 156, 752 – 768.

Kontoravdis, G., Bard, J.F., 1995. A grasp for the vehicle routing problem with time windows. ORSA Journal on Computing 7, 10–23. doi:10.1287/ijoc.7.1.10.

Lee, Y.H., Kang, J., Ryu, K.R., Kim, K.H., 2005. Optimization of container load sequencing by a hybrid of ant colony optimization and tabu search, in: Lipo Wang, Ke Chen, Y.S.O. (Ed.), Advances in Natural Computation. Springer, pp. 1259–1268.

Legato, P., Trunfio, R., Meisel, F., 2012. Modeling and solving rich quay crane scheduling problems. Computers & Operations Research 39, 2063 – 2078.

Li, C.L., Vairaktarakis, G.L., 2004. Loading and unloading operations in container terminals. IIE Transactions 36, 287–297.

Meisel, F., Wichmann, M., 2010. Container sequencing for quay cranes with internal reshuffles. OR spectrum 32, 569–591.

Monaco, M.F., Sammarra, M., Sorrentino, G., 2014. The terminal-oriented ship stowage planning problem. European Journal of Operational Research 239, 256–265.

Pacino, D., Delgado, A., Jensen, R.M., Bebbington, T., 2011. Fast generation of near-optimal plans for eco-efficient stowage of large container vessels, in: Computational Logistics. Springer, pp. 286–301.

Pardalos, L., Resende, M., 1994. A greedy randomized adaptive search procedure for the quadratic assignment problem. Quadratic Assignment and Related Problems, DIMACS Series on Discrete Mathematics and Theoretical Computer Science 16, 237–261.

Parreno, F., Pacino, D., Alvarez-Valdes, R., 2016. A grasp algorithm for the container stowage slot planning problem. Transportation Research Part E: Logistics and Transportation Review 94, 141 – 157.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40, 455–472.

Steenken, D., Voß, S., Stahlbock, R., 2004. Container terminal operation and operations research-a classification and literature review. OR Spectrum 26, 3–49.

Steenken, D., Winter, T., Zimmermann, U.T., 2001. Stowage and transport optimization in ship planning, in: Martin Groetschel, Sven O. Krumke, J.R. (Ed.), Online optimization of large scale systems. Springer, pp. 731–745.

Turkogullari, Y.B., Taskin, Z.C., Aras, N., Altinel, I.K., 2016. Optimal berth allocation, time-variant quay crane assignment and scheduling with crane setups in container terminals. European Journal of Operational Research 254, 985 – 1001.

UNCTAD, S., 2016. Review of maritime transport 2016, in: United Nations Conference on Trade and Development.

Zeng, Q., Yang, Z., 2009. Integrating simulation and optimization to schedule loading operations in container terminals. Computers & Operations Research 36, 1935 – 1944.

Zhen, L., 2016. Modeling of yard congestion and optimization of yard template in container ports. Transportation Research Part B: Methodological 90, 83 – 104.

Zhen, L., Wang, S., Wang, K., 2016a. Terminal allocation problem in a transshipment hub considering bunker consumption. Naval Research Logistics (NRL) 63, 529–548.

Zhen, L., Xu, Z., Wang, K., Ding, Y., 2016b. Multi-period yard template planning in container terminals. Transportation Research Part B: Methodological 93, 700 – 719.

A. Container swapping: Speed up

As mentioned in Section 6.2.4, calculating the improvement from performing a swap is the most computationally expensive part of the container swapping improvement method. This appendix describes how costs calculated previously can be reused, thus speeding up the overall method. Algorithm 3 describes how the improvement of swapping two containers (in terms of objective value) is calculated.

Consider two containers c_1 and c_2 their respective position in the current solution $p(c_1)$, $p(c_2)$, and their QCs $q(c_1)$ and $q(c_2)$. First observe that if the swap implies no change in the driving times, then the objective will not change due

Algorithm 3 `EvaluateSwapImprovement`

Input: x, c_1, c_2

1: **if** $\tau_{c_1,p(c_1)} = \tau_{c_2,p(c_1)}$ **and** $\tau_{c_2,p(c_2)} = \tau_{c_1,p(c_2)}$ **then**
2: **if** $q(c_1) = q(c_2)$ **then**
3: `swapAndUpdateTimes`(x, c_1, c_2)
4: workSaved $\leftarrow$ `calculateWorkSaved`(x, c_1, c_2)
5: newFinishTime $\leftarrow \max\left(\texttt{calculateFinishTime}(x, c_1, c_2), \max_{q \in Q \setminus \{q(c_1)\}} (\texttt{craneFinishTime}(x, q))\right)$
6: lateSaved $\leftarrow$ `calculateLateSaved`$(x,$ newFinishTime$)$
7: **return** α·workSaved$+\gamma$·lateSaved
8: **else**
9: workSaved$_1$, newFinishTime$_1$ $\leftarrow$ `saveSwapProfit`$(x, p(c_1), \tau_{c_2,p(c_1)}, c_1, c_2)$
10: workSaved$_2$, newFinishTime$_2$ $\leftarrow$ `saveSwapProfit`$(x, p(c_2), \tau_{c_1,p(c_2)}, c_2, c_1)$
11: workSaved $\leftarrow$ workSaved$_1$ + workSaved$_2$
12: newFinishTime $\leftarrow \max\left(\text{newFinishTime}_1, \text{newFinishTime}_2, \max_{q \in Q \setminus \{q(c_1), q(c_2)\}} (\texttt{craneFinishTime}(x, q))\right)$
13: lateSaved $\leftarrow$ `calculateLateSaved`$(x,$ newFinishTime$)$
14: **return** α·workSaved$+\gamma$·lateSaved
15: **end if**
16: **else**
17: **return** 0
18: **end if**

to the swap (lines 1, 16-17). If the driving times are different, the algorithm considers two cases; $q(c_1)$ and $q(c_2)$ are the same, or they are different (lines 2 and 8).

In the case when the two considered QCs are identical, the improvement is calculated by swapping the containers and iteratively updating the delivery times for the container-assignments (line 3). With the times updated the work saved and the reduction in the tardiness is easily computed by comparing with the solution x (lines 4-6).

In the case when the two considered QCs are different, the cost calculation is done container for container (lines 9-10). The procedure `saveSwapImprovement`(x, p, τ, c_1, c_2) takes two containers as input and returns the work saved and the new finishing time for QC $q(c_1)$ when swapping containers c_1 and c_2. Hereafter the cost improvement is calculated similarly to the previous case (lines 11-13).

The fundamental idea behind `saveSwapImprovement`(x, p, τ, c_1, c_2) is that you can compute the costs when needed, and reuse these costs when appropriate instead of calculating it again. Keep in mind that this swapping method does not put a position on a new TV, but simple changes the container to be loaded on a position to another compatible container. The important thing here is the driving time to the new container. Consider the following situation; you have two candidates swaps $\{c_1, c_2\}$ and $\{c_1, c_3\}$ where $q(c_1) \neq q(c_2)$ and $q(c_1) \neq q(c_3)$. The changes in the cost for QC $q(c_1)$ only depends on the driving time to the new container. If the driving time from position $p(c_1)$ to container c_2 is the same as the driving time to c_3, then you know the cost for QC $q(c_1)$ will be the same for these two swaps. Therefore you really only need to calculate the cost once, store it and reuse it for the second case. This is exactly what is done in `saveSwapImprovement`(x, p, τ, c_1, c_2) as seen in Algorithm 4.

Algorithm 4 `saveSwapImprovement`

Input: x, p, τ, c_1, c_2

1: workSaved $\leftarrow \mathbf{W}(c_1, \tau)$
2: newFinishTime $\leftarrow \mathbf{F}(c_1, \tau)$
3: **if** workSaved $= \emptyset$ **then**
4: `swapAndUpdateTimes`(x, c_1, c_2)
5: workSaved $\leftarrow$ `calculateWorkSaved`(x, c_1, c_2)
6: newFinishTime $\leftarrow$ `calculateFinishTime`(x, c_1, c_2)
7: $\mathbf{W}(c_1, \hat{\tau}) \leftarrow$ workSaved
8: $\mathbf{F}(c_1, \hat{\tau}) \leftarrow$ newFinishTime
9: **end if**
10: **return** workSaved, newFinishTime

The costs, as well as the finishing time for QC $q(c_1)$, are stored in $\mathbf{W}(c_1, \tau)$ and $\mathbf{F}(c_1, \tau)$. If we have not calculated the cost yet we calculate the cost and store it (lines 3-8), otherwise, we reuse what we have calculated (lines 1-2). $\mathbf{W}$ and $\mathbf{F}$ are stored globally and can be reused between moves within the improvement method. Only when making changes to a QC q do we have to reset the costs for all containers using QC q, as these costs are no longer ensured to be valid. When doing so the cost for picking up a container with the same driving time as the current one is set to 0, and the finish time is set to the finishing time for QC q

$$\mathbf{W}(c, \tau_{c,p(c)}) = 0 \qquad \forall c \in \{c' \in C \mid q(c') = q\}$$
$$\mathbf{F}(c, \tau_{c,p(c)}) = \texttt{craneFinishTime}(x, q) \qquad \forall c \in \{c' \in C \mid q(c') = q\}$$