Knowledge-Based Systems 29 (2012) 63–74

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A decision support system for managing combinatorial problems in container terminals

Miguel A. Salido [*], Mario Rodriguez-Molins, Federico Barber

Instituto de Automática e Informática Industrial, Universidad Politécnica de Valencia, Valencia, Spain

ARTICLE INFO

Article history:
Available online 23 July 2011

Keywords:
Berth allocation problem
Quay Crane Assignment Problem
Container Stacking Problem
Decision support system
Artificial intelligence
Metaheuristic techniques

ABSTRACT

A container terminal is a facility where cargo containers are transshipped between different transport vehicles. We focus our attention on the transshipment between vessels and land vehicles, in which case the terminal is described as a maritime container terminal. In these container terminals, many combinatorial related problems appear and the solution of one of the problems may affect to the solution of other related problems. For instance, the berth allocation problem can affect to the crane assignment problem and both could also affect to the Container Stacking Problem. Thus, terminal operators normally demand all containers to be loaded into an incoming vessel should be ready and easily accessible in the yard before vessel's arrival. Similarly, customers (i.e., vessel owners) expect prompt berthing of their vessels upon arrival. However the efficiency of the loading/unloading tasks of containers in a vessel depends on the number of assigned cranes and the efficiency of the container yard logistic. In this paper, we present a decision support system to guide the operators in the development of these typical tasks. Due to some of these problems are combinatorial, some analytical formulas are presented to estimate the behavior of the container terminal.

1. Introduction

Container terminals generally serve as a transshipment between ships and land vehicles (trains or trucks). Henesey shows in [17] how this transshipment market is growing fast. Between 1990 and 2008, container traffic has grown from 28.7 million to 152.0 million of movements. This corresponds to an average annual compound growth of 9.5%. In the same period, container throughput went from 88 million to 530 million of containers, which represents an increase of 500%. The surge of both container traffic and throughput is linked with the growth of international trade in addition to the adoption of containerization as privileged vector for maritime shipping and inland transportation [1].

The efficient management of containers in port requires more analysis and development to ensure reliability, delivery dates or handling times in order to improve productivity and container throughput from quay to landside and vice versa. Extensive surveys are provided about operations at seaport container terminals and methods for their optimization [32,30]. Moreover, other problems are faced on planning the routes for liner shipping services to obtain the maximal profit [7]. Another important issue for the success at any container terminal is to forecast container throughput accurately [5]. Thus, they could develop better operational strategies and investment plans.

The main research on optimization methods in container terminals is related to reduce the berthing time of vessels. This objective generates a set of interrelated problems such as berth allocation, yard-side operation, storage operation and gatehouse operation. Usually, each one of these problems is managed independently of others due to their exponential complexity. However, these problems are clearly interrelated so that an optimized solution of one of them restrings the possibility of obtaining a good solution in another.

The overall goal collaboration between our group at the Technical University of Valencia (UPV), Valencia Port Foundation, and the maritime container terminal MSC (Mediterranean Shipping Company S.A.) is to offer assistance to help in planning and scheduling tasks such as the allocation of spaces to outbound containers, to identify bottlenecks, to determine the consequences of changes, to provide support in the resolution of incidents, to provide alternative berthing plans, etc.

In this paper, we focus our attention on three important and interrelated problems: the berth allocation problem (BAP), the Quay Crane Assignment Problem (QCAP) and the Container Stacking Problem (CStackP) (see Fig. 1). Briefly, the BAP and QCAP consist of the allocation of docks and quay cranes to incoming vessels under several constraints and priorities (length and depth of vessels, number of containers, etc.). On the other hand, when a

* Corresponding author. Tel.: +34 963877000x83512; fax: +34 963877359.
E-mail address: msalido@dsic.upv.es (M.A. Salido).

0950-7051/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.06.021

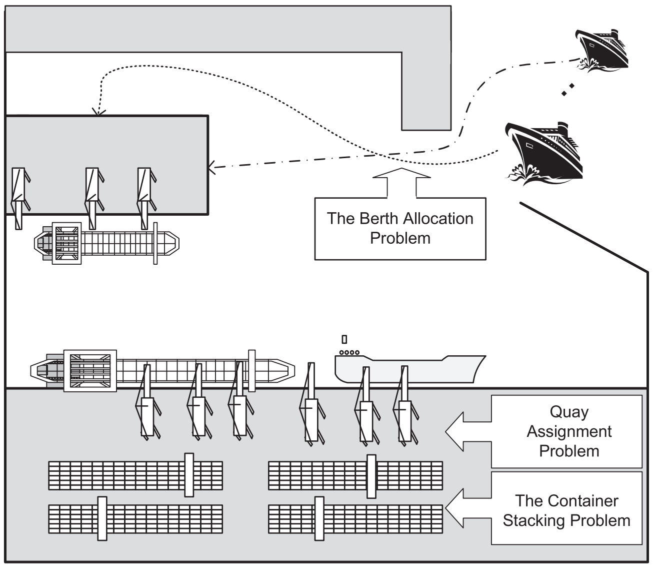


Fig. 1. Integrated remarshaling, berthing and quay crane allocation problems in maritime terminals.

vessel berths, export containers stacked to be loaded in the vessel should be on top of the stacks of the container yard. Therefore, the CStackP consists of relocating the containers so that the yard crane does not need to do re-handling work at the time of loading. These two problems are clearly related: an optimal berth allocation plan may generate a large amount of relocations for export containers; meanwhile a suboptimal berth allocation plan could require fewer rearrangements. Terminal operators should decide which solution is the most appropriate in each scenario.

In order to provide a computer-based decision support system, we integrate a set of intelligent techniques for solving these problems concurrently in order to achieve a mixed-solution that combines optimization of BAP, QCAP and CStackP. To this end, we developed a heuristically-guided planner for generating a rehandling-free intra-block remarshaling plan for container yards (CStackP problem). Due to the fact that this is a time consuming task, we present in this paper an analytic formula to estimate the number of reshuffles needed to solve this problem. Then, we present a meta-heuristic approach for solving the BAP + QCAP as an independent problem. Afterwards, we integrate solutions obtained from BAP + QCAP and StackP systems, so that terminal operators should ultimately decide which solution is the most appropriate in relation to a multi-objective function: to minimize the waiting times of vessels and to minimize the amount of relocations of containers.

These techniques will be very useful for terminal operators due to berth allocation is especially important in case of ship delays because in this case a new berthing place has to be allocated to the ship whereas containers are already stacked in the yard [30] and a remarshaling plan remains necessary to minimize the berthing time.

2. Integrating BAP, QCAP and CStackP

As we have pointed out, both the CStackP and the BAP + QCAP are well-known problems and several techniques have been developed to solve them separately. However, few systems have been developed to relate and optimize both problems in an integrated way. Some works consider berth and yard planning in a common optimization model [2,4,10], but they are mainly focused on storage strategies. Moreover, only some works integrate the BAP with the QCAP. Giallombardo et al. [12] try to minimize the yard-related house-keeping costs generated by the flows of containers exchanged between vessels. However, there also exists a relationship between the optimization of maritime and terminal-sides operations (BAP, QCAP, CStackP, etc.). Fig. 2 shows an example of three berth allocation plans with the corresponding quay crane allocations and a block of containers to be loaded in the vessels. Containers of type A, B and C must be loaded in vessels A, B and C, respectively. In the first berth allocation plan, the order of vessels is A–B–C and the quay crane allocation is two cranes, three cranes and one crane, respectively. The second berth allocation plan is C–B–A. In this case the quay crane allocation is three, two and one, respectively. Finally, the third berth allocation plan is B–C–A and two quay cranes are allocated to all vessels. Each configuration generates a different waiting time for berthing and different handling times, and the port operator probably selects the best solution to optimize these (BAP and QCAP) problems. However the best solution of these two problems could generate a large number of reshuffles in the yard so the question is straightforward: what is a better solution? Perhaps a solution that optimizes the BAP + QCAP could not be the more appropriate for the CStackP (and vice versa).

Given a waiting queue of vessels to be allocated and a given state of the containers in the container yard, each solution for the BAP + QCAP ($SBAP_i$: a feasible sequence of mooring and a feasible quay crane allocation), requires a different number of container's re-locations in the associated CStackP solution ($SCStackP_i$) in order to put on top the containers to be loaded according to the order of berthing. We can associate a cost to each $SBAP_i$ + $SQCAP_i$ related to the total weighted waiting time and handling time of vessels of this berthing order (T_w). Likewise, we can associate a cost to each $SCStackP_i$ as the number of required container relocations. Therefore, we can qualify the optimality of each global solution (Sol_i) of BAP + QCAP and CStackP as a lineal combination of the quality of each partial solution:

$$Cost(Sol_i) = \alpha * Cost(SBAP_i + SQCAP_i) + \beta * (SCStackP_i) \quad (1)$$

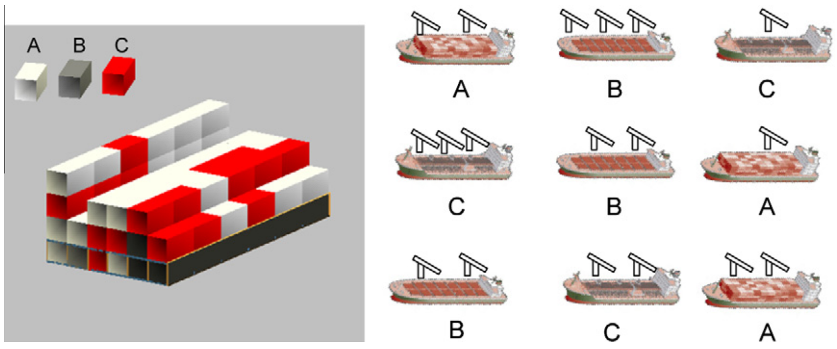


Fig. 2. Different alternatives of BAP and QCAP.

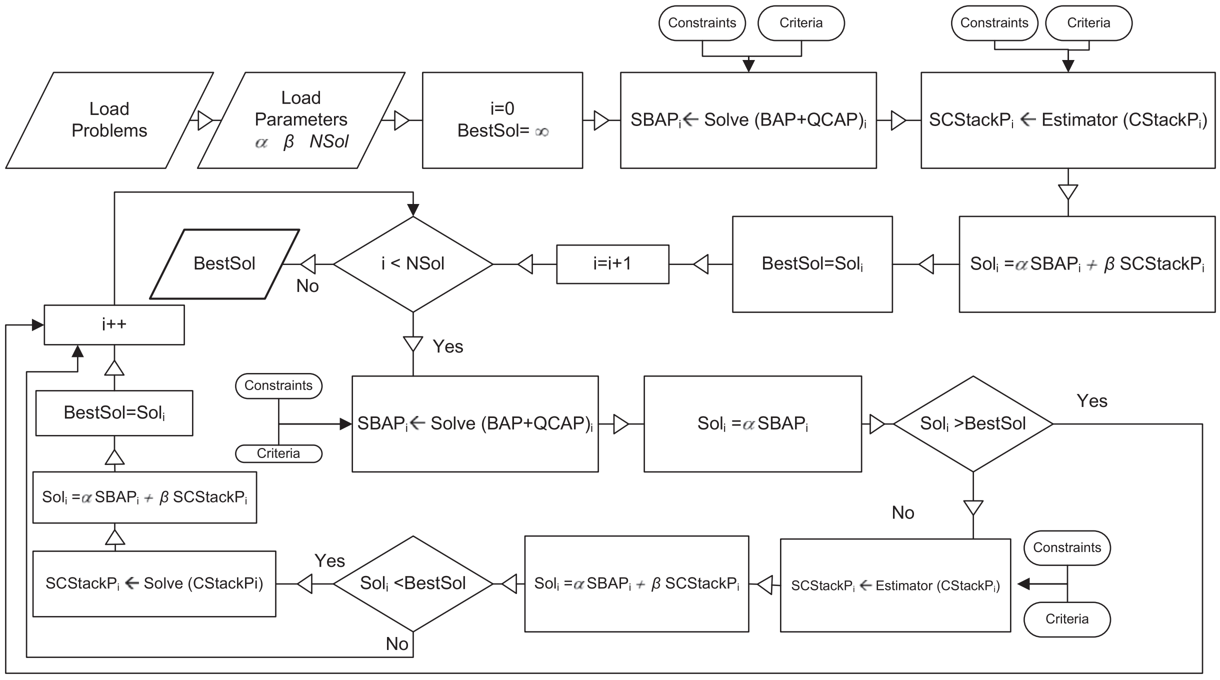


Fig. 3. Data flow diagram of the Integrated System Functioning.

The best decision will depend on the policy of each maritime terminal (α and β parameters). The data flow diagram of the Integrated System Functioning can be seeing in Fig. 3. Firstly, the BAP, QCAP and the CStackP data are loaded in the integrated system. Next, the BAP + QCAP is solved to achieve a solution ($SBAP_i$) based on their constraints and optimization criteria. Then, the CStackP is estimated by taking into account the berthing order of vessels obtained in $SBAP_i$. This estimator returns the number of reshuffles needed to achieve a solution. After this step, the cost of the global solution (Sol_i) can be calculated by using the previous expression (Eq. (1)). By iterating this integrated process, the operators can obtain a qualification cost of each feasible Sol_i, as well as the best global solution, according to the given α and β parameters. A branch and bound method has been also applied in the integrated search for the best global solution (Sol_i), so that the search can be pruned each time the current solution does not improve the best solution found so far. Finally, once the best solution is obtained, the CStackP planner is executed to obtain the specific movements for all remarshaling tasks. This plan is sequentially obtained for each vessel according to the solution obtained in $SBAP_i$ and the current state of the container yard. Thus, the optimized remarshaling plan for the berthing order of vessels of $SBAP_i$ is obtained.

In next sections we develop our proposed techniques for estimating and solving the Container Stacking Problem, the berth allocation problem and the quay crane allocation problem in order to achieve a global solution Sol_i to the integrated problems.

3. The Container Stacking Problem

Containers are ISO standardized metal boxes which can be stacked on top of each other. A container yard (see Fig. 4) is composed of several blocks, each one consisting of (20–30) yard-bays. Each yard-bay contains several (usually 6) rows and each row has a maximum allowed tier (usually 4 or 5 tiers for full containers).

Loading and offloading containers on the stack is performed by cranes following a ‘last-in, first-out’ (LIFO) criteria. Containers are stacked in the order they arrive. However, in the loading process

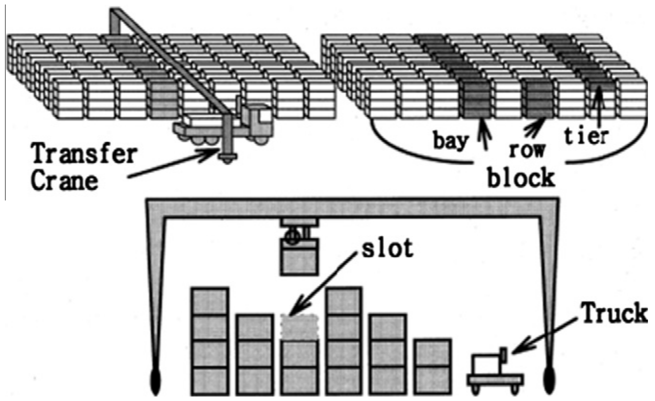


Fig. 4. A container yard.

of vessels, to access a container which is not at the top of its pile, those above it must be relocated. This remarshaling process is required since the stacking order depends on the order in which ships unload or containers have been stacked. This remarshaling process reduces the productivity of cranes and its optimization would minimize the moves required. For safety reasons, it is usually prohibited to move the gantry crane while carrying a container [22], therefore these movements only take place in the same yard-bay. In addition, there exist a set of hard/soft constraints regarding container moves or locations where can be stacked, for example, small differences in height of adjacent yard-bays, dangerous containers must be allocated separately by maintaining a minimum distance, etc. The CStackP is a NP-complete combinatorial optimization problem and different approaches have been proposed ([27,20], etc.). The CStackP can be viewed, from the artificial intelligence point of view, as a modification of the Blocks World planning domain [33].

In [28], a planning system for remarshaling processes was proposed. This system obtains the optimized plan of reshuffles of containers in order to allocate all selected containers at the top of the stacks, or under another selected containers, in such a way that no reshuffles will be needed to load these outgoing containers. This planner was specified by means of the standard planning domain definition language (PDDL) [11] and it was developed on the well-known domain-independent planner MetricFF [18]. The developed domain file contains the common features of the problem domain: (i) the domain objects: containers and rows, (ii) the relations among them (propositions), and (iii) allowed moves to change the status of the problem (actions). The problem file describes each particular instance: (i) the initial layout of the containers in the yard (Initial state), (ii) the export containers (goal) which must be allocated at the top of the stacks or under other export containers, and (iii) the function to optimize (minimizing the number of relocation movements). Fig. 5 shows a simple example of a problem instance in PDDL. This problem describes a scenario with three containers (X1, X2 and X3) in the yard-bay and the crane is not holding any of them. X2 is an export container which must be relocated in order to be easily accessible at loading time of the next vessel.

In [29] the Metric-FF-based initial planner was improved by integrating a domain-dependent heuristic (H1) in order to achieve efficiency. H1 computes an estimator of the number of container movements that must be carried out to reach a goal state, which it is used to guide search of solutions. However, new constrains and optimization criteria were included in order to take into account real-world requirements:

C.1. Reducing distance of the outgoing containers to the cargo side (to the left or right hand side of bays).
C.2. Increasing the range of the move actions set for the cranes allowing moving a container to 5th tier (in yard-bays with tiers 4).
C.3. Balancing the number of stacked containers within the same bay in order to avoid sinks, that is, the difference between the number of containers stacked in adjacent positions should be limited.

This planner was improved to manage a full container yard by distributing in small problems. On other issues such as monitoring, distributed approaches have been used [24]. The container yard was decomposed in yard-bays, so that the problem was distributed into a set of subproblems. Thus, each yard-bay generated a subproblem. However, containers of different yard-bays must satisfy a set of constraints among them. Therefore, subproblems were sequentially solved, so that each subproblem (yard-bay) took into account the set of constraints with previously solved subproblems. This decomposition required taking into account these new added constraints. With these new added constraint and criteria, the developed planner could solve more real-world based problems:

1. Balancing contiguous yard-bays: rows of adjacent yard-bays must be balanced in order to avoid sinks inter yard-bays (CB).
2. Dangerous containers must maintain a minimum security (Euclidian) distance among them (DC).

Due to we manage the entire container yard, the requirement C.3 must be extended to balance the stacks of contiguous yard-bays. Thus the planning of a yard-bay is carried out taking into account the solution obtained by the previous yard-bay.

3.1. An ordered yard-based planner: Planner H1 Ordered

In order to insert our planner in the integrated system, we have also improved our version to minimize the number of reshuffles for a set of outgoing containers to be loaded in successive berthed vessels. Initially our planner was developed to minimize the number of reshuffles for one vessel (vessel A in Fig. 6). However, the order of the rest of containers in the yard-bay did not matter. The new planner (Planner H1 Ordered) takes into account these features and it is able to organize the bay in order to adapt to the berth schedule. Thus, the reshuffles needed to allocate the outgoing containers for a vessel are carried out taking into account the outgoing containers for the following vessel to berth.

In the previous version of our planner (Planner H1 Sequential), the order of execution of yard-bays was sequential. Thus a first plan was obtained for the first yard-bay. Then a new plan is carried out for the second yard-bay taking into account the balance constraints generated with the solution obtained for the first yard-bay, and so on.

The new version of our planner classifies the yard-bays by means of tightness. Fig. 6 shows the order of execution of a complete yard. A natural order of execution is the sequential order, from the first yard-bay to the end yard-bay (Planner H1 Sequential).

However, in this version (Planner H1 Ordered), the natural order of execution is modified. The tightest yard-bays (yard-bays with more export containers) are solved first. Thus, the number

```
(define (problem p1) (:domain CStackP)

 ;;Domain objects
 (:objects  X1 X2 X3 - container S1 S2 S3 S4 S5 S6 - slot )

 ;;Initial state
 (:init
(clear X1)(clear X3)(clear S1)(clear S4)(clear S5)(clear S6)
(on X2 S2)(on X1 X2)(on X3 S3)(at X1 S2)(at X2 S2)(at X3 S3)
(= (height S1) 0)(= (height S2) 2)(= (height S3) 1)
(= (height S4) 0)(= (height S5) 0)(= (height S6) 0)
(= (num-moves) 0)(handempty)
(goal-container X2)
 )

 ;;Goal state
 (:goal (and (handempty) (ready X2)) )

 ;;function to optimize
 (:metric minimize (num-moves))
)
```

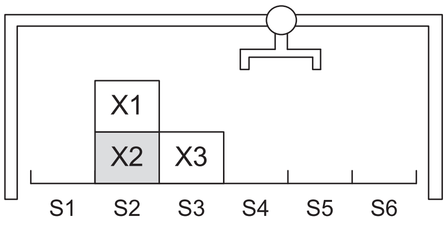


Fig. 5. Example of problem instance in Container Stacking domain.

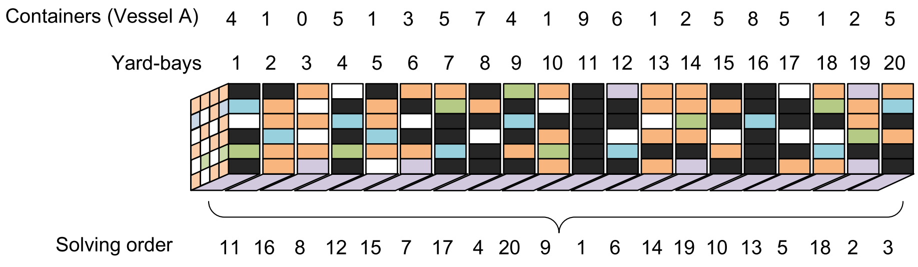

Containers (Vessel A) 4 1 0 5 1 3 5 7 4 1 9 6 1 2 5 8 5 1 2 5
Yard-bays 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Solving order 11 16 8 12 15 7 17 4 20 9 1 6 14 19 10 13 5 18 2 3

Fig. 6. Order of execution of yard-bays.

of reshuffles is minimized due to the fact that the tightest yard-bays are solved without the need of satisfying the balance constraints with the contiguous ones, meanwhile their neighbor yard-bays must be committed to these tasks. Thus, following the example of Fig. 6, the yard-bay 11 is executed first without balance constraints because it has 9 goal containers meanwhile yard-bays 10 and 12 are executed later taking into consideration the balance constraints generated by the solution of yard-bay 11. In the evaluation section we will compare the behavior of this planner against the previous version. In general the order of execution of yard-bays is directly related with the efficiency of our planning tool. The ordering of yard-bays by tightness improves the efficiency of our planning tool.

3.2. An analytic formula to estimate the number of reshuffles

As we have pointed out, solving the CStackP is a NP-complete combinatorial optimization problem. Once the BAP returns a possible schedule of vessels to berth in the port, the plan of yard reshuffles must be carried out for each vessel. This is a very hard task so that a general formula that estimates the number of reshuffles for each vessel remains necessary. Once the global solution is achieved, the planning tool is executed to obtain the optimal plan for the yard reshuffles. To this end, we have identified the main parameters that affect the number of reshuffles in a yard-bay for a vessel:

1. Number of total slots in a yard-bay (P_1). For instance, a yard-bay with tier 5 and 6 rows, the number of slots is 30.
2. Number of current containers in the yard-bay (P_2), $P_2 \leqslant P_1$.
3. Number of goal (outgoing) containers (P_3), $P3 \leqslant P2$
4. Number of containers on top of goal containers (P_4). A lower bound for the minimal number of reshuffles is P_4.

These parameters influence in the number of reshuffles needed for each yard-bay. The estimator R is mainly depended on P_4 and it is bounded by:

$$R = P_4 + \alpha : \alpha \in [0, \infty) \qquad (2)$$

where $\alpha = 0$ means that the problem is underconstrained meanwhile $\alpha \to \infty$ means that the problem is unsolvable.

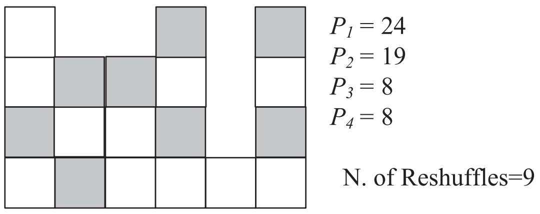


Fig. 7. Example and values of parameters. Grey containers are goal containers.

Fig. 7 shows an example of yard-bay with the value of each identified parameter. The rest of parameters also play an important role but the relationship among them is secondary. The simulation of one hundred yard-bays with different tiers and number of containers and objectives return a strong relationship among the parameters P_1, P_2 and P_3. If $(P_1 - P_2)/P_3$ is lower than 1.25, one more reshuffle is needed $(P_4 + 1)$. In the same way if $(P_1 - P_2)/P_3$ is lower than 1, one more reshuffle is also needed $(P_4 + 2)$; and so on. Finally if $(P_1 - P_2) \to 0$ the problem is unsolvable. Thus, we estimate the number of reshuffles by the following formula:

$$R = P_4 + \left\lfloor \frac{2}{\frac{P_1 - P_2}{P_3}} \right\rfloor = P_4 + \left\lfloor \frac{2 \cdot P_3}{P_1 - P_2} \right\rfloor \qquad (3)$$

This estimator R is accurate enough for us to do not need to execute our planner for each berthing plan. In the integrated system presented above we can select to run our planner or to use the estimator. In any case, once the best solution is found for the integrated problem, our planner must solve the best solution in order to determine the specific plan which will be carried out by the cranes to allocate the containers in the appropriate places.

4. The berth allocation and Quay Crane Assignment Problem

We will focus our attention on the berth allocation problem (BAP), a well-known NP-Hard combinatorial optimization problem, which consists of assigning incoming vessels to berthing positions. Once a vessel arrives at the port (see Fig. 8), it enters in the harbor waiting time to moor at the quay. The quay is a platform protruding into the water to facilitate the loading and unloading of cargo. The locations where mooring can take place are called berths. These are equipped with giant cranes, called pier or quay cranes (QC), used to load and unload containers which are transferred to and from the yard by a fleet of vehicles. In a transshipment terminal the yard allows temporary storage before containers are transferred to another ship or to another mode (e.g., rail or road).

The BAP is one of the most relevant problems arising in the management of container ports. Several models are usually considered [31].

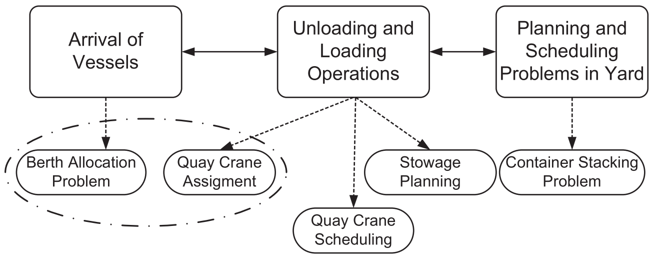


Fig. 8. Planning and scheduling problems in container terminals.

Managers at container terminals face two interrelated decisions: *where* and *when* the vessels should moor. First, they must take into account physics restrictions as length or draft, but also they have to take into account the priorities and other aspects to minimize both port and user costs, which are usually opposites. Fig. 9 shows an example of graphical space–time representation of a berth planning with 6 vessels. Each rectangle represents a vessel with its service time and length.

In [31], the authors show a complete comparative study about different solutions for the BAP according to their efficiency in addressing key operational and tactical questions relating to vessel service. They also study the relevance and applicability of the solutions to the different strategies and contractual service arrangements between terminal operator and shipping lines.

To show similarities and differences in the existing models for berth allocation, Bierwirth and Meisel [3] developed a classification scheme (see Fig. 10). They classify the BAP according to four attributes. The spatial attribute concerns the berth layout and water depth restrictions. The temporal attribute describes the temporal constraints for the service process of vessels. The handling time attribute determines the way vessel handling times are considered in the problem. The fourth attribute defines the performance measure to reflect different service quality criteria. The most important ones are minimizing the waiting time and the handling time of a vessel. Both measures aim at providing a competitive service to vessel operators. If both objectives are pursued (i.e. wait and hand are set), the port stay time of vessels is minimized. Other measures are focused on minimizing the completion times of vessels among others. Thus, by using the above classification scheme, a certain type of BAP is described by a selection of values for each of the attributes. For instance, a problem where the quay is assumed to be a continuous line (*cont*). The arrival times restrict the earliest berthing of vessels (*dyn*) and handling times depends on the berthing position of the vessel (*pos*). The objective is to minimize the sum of the waiting times (*wait*) and handling time (*hand*). According to the scheme proposed by Bierwirth and Meisel [3], this problem is classified by $cont|dyn|pos|\Sigma(wait + hand)$.

One of the early works that appeared in the literature was [21], in which they developed a heuristic algorithm by considering a First-Come-First-Served (FCFS) rule. However, the idea that for high port throughput, optimal vessel-to-berth assignments should be found without considering the FCFS bases was introduced by Imai et al. [19]. Therefore, we will use the FCFS rule in order to get an upper bound of the function cost in BAP. Nevertheless, this approach may result in some vessels' dissatisfaction regarding the order of service.

In [13], multiple vessel mooring per berth are allowed assuming that vessel arrivals can be grouped into batches. They have developed a tree search procedure which provides an exact solution and this is improved by a composite heuristic.

Metaheuristics have been developed to solve the BAP but have also been used in other fields such as flowshop or flexible job-shop scheduling problems [25,14]. On the one hand, Cordeau et al. [8] introduce two Tabu Search heuristics to solve the discrete and continuous case, respectively to minimize is the weighted sum for every ship of the service time in the port. Both heuristics are inspired by a *Multi-Depot Vehicle Routing Problem with Time Windows* algorithm and can handle various features of real-life problems as time windows or favorite and acceptable berthing areas. Mauri et al. [23] design a column generation approach for the problem of Cordeau et al. [8] which delivers better solutions in shorter runtime than Tabu Search. In the models of Han et al. [15] and Zhou et al. [34], a Genetic Algorithm (GA) is proposed to solve the problem. In both models the draft of vessels restricts the berth assignment decisions.

An approach based on multi-objective optimization problem using evolutionary algorithms [6] is followed to minimize the makespan of the port, total waiting time of the ships, and degree of deviation from a predetermined service priority schedule.

In [12], they present the integration of BAP with the Quay Crane Assignment Problem (QCAP) through two mixed integer programming formulations including a tabu search method which is an adaption of the one of [8], however they minimize the yard-related housekeeping costs generated by the flows of containers exchanged between vessels. In [3], the authors give a

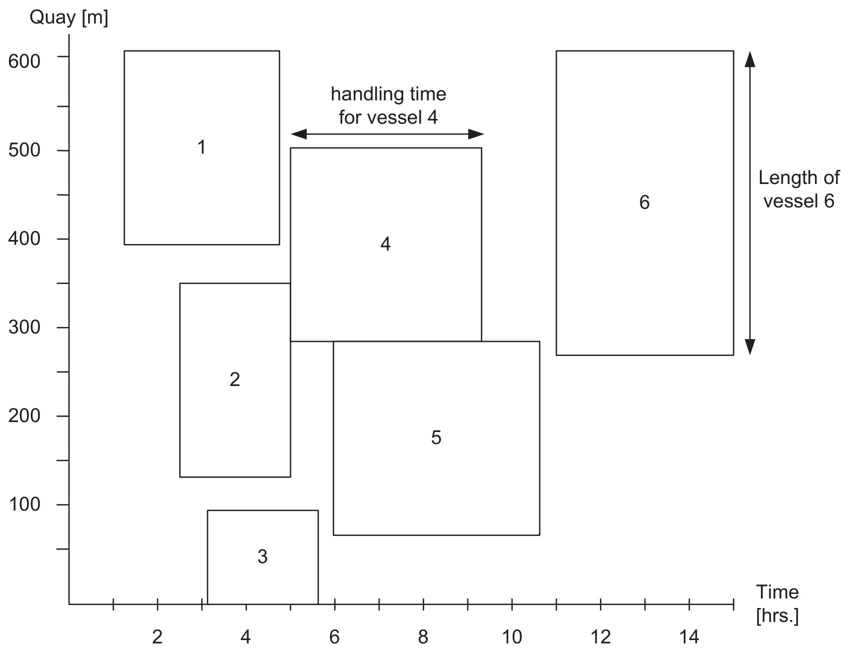
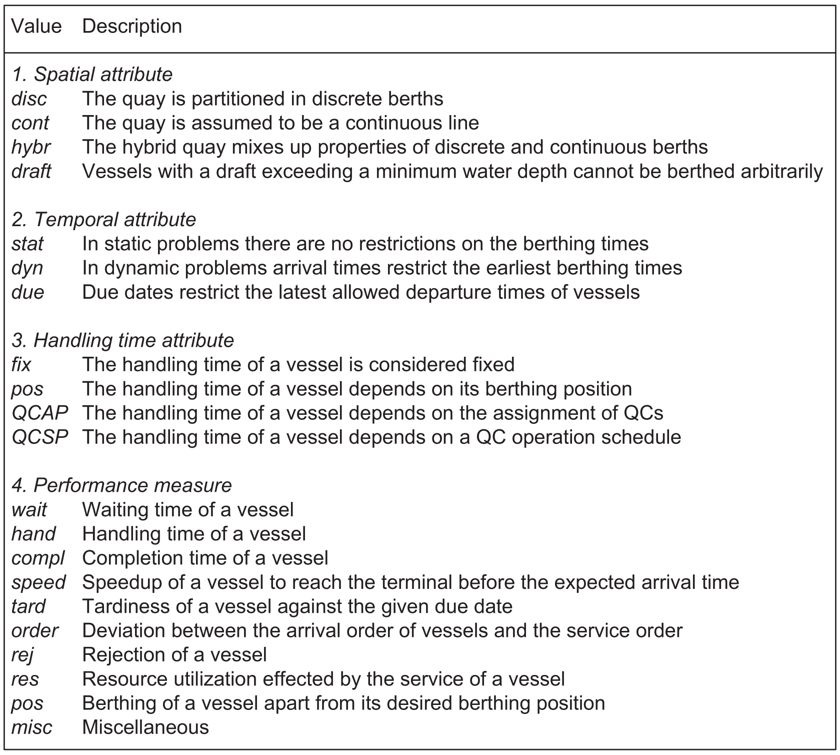


Fig. 9. A berth planning.



Value	Description
1. Spatial attribute	
disc	The quay is partitioned in discrete berths
cont	The quay is assumed to be a continuous line
hybr	The hybrid quay mixes up properties of discrete and continuous berths
draft	Vessels with a draft exceeding a minimum water depth cannot be berthed arbitrarily
2. Temporal attribute	
stat	In static problems there are no restrictions on the berthing times
dyn	In dynamic problems arrival times restrict the earliest berthing times
due	Due dates restrict the latest allowed departure times of vessels
3. Handling time attribute	
fix	The handling time of a vessel is considered fixed
pos	The handling time of a vessel depends on its berthing position
QCAP	The handling time of a vessel depends on the assignment of QCs
QCSP	The handling time of a vessel depends on a QC operation schedule
4. Performance measure	
wait	Waiting time of a vessel
hand	Handling time of a vessel
compl	Completion time of a vessel
speed	Speedup of a vessel to reach the terminal before the expected arrival time
tard	Tardiness of a vessel against the given due date
order	Deviation between the arrival order of vessels and the service order
rej	Rejection of a vessel
res	Resource utilization effected by the service of a vessel
pos	Berthing of a vessel apart from its desired berthing position
misc	Miscellaneous

Fig. 10. A classification scheme for BAP formulation [3].

comprehensive survey of berth allocation and quay crane assignment formulations from the literature. Some authors outline approaches more or less informally while others provide precise optimization models. More than 40 formulations are presented distributed among discrete problems, continuous problems and hybrid problems. Hansen et al. [16] considered a discrete problem with a tardiness objective which accounts for departure time related costs including penalties for tardiness as well as benefits for early departures. This problem was solved by a variable neighborhood search which turns out to be superior to the GA of Nishimura et al. [26].

Nowadays, this process is generally solved manually and it is usually solved by means of a policy to serve the first vessel arrives.

Considering the requirements of container operators of MSC (Mediterranean Shipping Company S.A.), our approach also studies the integration of these two problems (BAP and QCAP) through a metaheuristic called Greedy Randomized Adaptive Search Procedures (GRASP) [9]. This metaheuristic is able to find feasible solutions within an acceptable computational time. In this way and following the above classification scheme (see Fig. 10), our approach is classified by $cont|dyn|QCAP|\Sigma wait$. Thus, we focused on the following attributes and performance measure:

- *Spatial attribute: cont*: We assume the quay is a continuous line, so that there is no partitioning of the quay and vessel can berth at arbitrary positions within the boundaries of the quay. It must be taken into account that for a continuous layout, berth planning is more complicated than for a discrete layout at the advantage of better utilizing quay space [3].
- *Temporal attribute: dyn*: We assume dynamic problems where arrival times restrict the earliest berthing times. Thus, fixed arrival times are given for the vessels, hence, vessels cannot berth before their expected arrival time.
- *Handling time attribute: QCAP*: We assume the handling time of a vessel depends on the assignment of QCs.
- *Performance measure: wait*: Our objective is to minimize the sum of the waiting time of all vessels.

The objective in BAP is to obtain an optimal distribution of the docks and cranes to vessels waiting to berth. Thus, this problem could be considered as a special kind of machine scheduling problem, with specific constrains (length and depth of vessels, ensure a correct order for vessels that exchange containers, assuring departing times, etc.) and optimization criteria (priorities, minimization of waiting and staying times of vessels, satisfaction on order of berthing, minimizing cranes moves, degree of deviation from a pre-determined service priority, etc.).

4.1. Notation for BAP and QCAP

Our approach follows an integration of the Quay Crane Assignment Problem (QCAP) and the BAP through the metaheuristic Greedy Randomized Adaptive Search Procedure (GRASP) [9] which is able to obtain optimized solutions in a very efficient way. Following, we introduce the used notation:

Q_C	Available QCs in the container terminal. These QCs are identical in terms of productivity in loading/discharging containers.
L	Total length of the berth in the container terminal.
$a(V_i)$	Arrival time of the vessel V_i at port.
$m(V_i)$	Moored time of V_i. All constraints must hold.
$pos(V_i)$	Berthing position where V_i will moor.
$c(V_i)$	Number of required movements to load and unload containers of V_i.

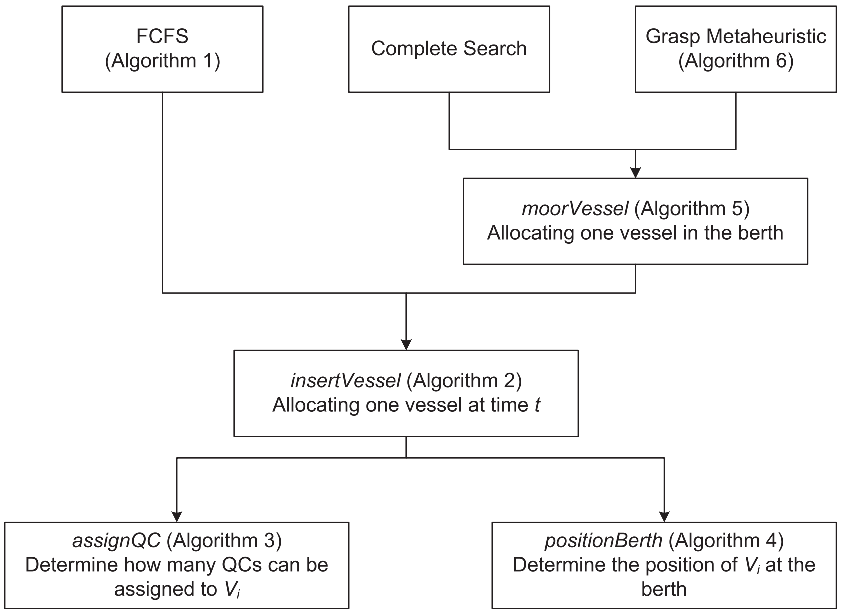

FCFS
(Algorithm 1)
Complete Search
Grasp Metaheuristic
(Algorithm 6)
moorVessel (Algorithm 5)
Allocating one vessel in the berth
insertVessel (Algorithm 2)
Allocating one vessel at time t
assignQC (Algorithm 3)
Determine how many QCs can be assigned to V_i
positionBerth (Algorithm 4)
Determine the position of V_i at the berth

Fig. 11. Application order of the algorithms presented.

$q(V_i)$ Number of assigned QCs to V_i. The maximum number of assigned QC by vessel depends on its length since a security distance is required between two contiguous QC (`secQC`) and the maximum number of cranes that the container terminal allows per vessel (`maxQC`). Let's assume that the number of QC does not vary along all the moored time. Thus, the handling time of V_i is given by (where `movsQC` is the QC's moves per unit time):

$$\frac{c(V_i)}{q(V_i) \times \mathtt{movsQC}} \tag{4}$$

$d(V_i)$ Departure time of V_i, which depends on $m(V_i)$, $c(V_i)$ and $q(V_i)$.

$w(V_i)$ Waiting time of V_i from it arrives at port until it moors: $w(V_i) = m(V_i) - a(V_i)$.

$l(V_i)$ Length of V_i. There is a distance security (`secLength`) between two moored ships: let's assume 5% of their lengths.

$pr(V_i)$ Vessel's priority.

In order to simplify the problem, let's assume that mooring and unmooring does not consume time and every vessel has a draft lower or equal than the quay. In each case, simultaneous berthing is allowed.

The goal of the BAP is to allocate each vessel according existing constraints and to minimize the total weighted waiting time of vessels:

$$T_w = \sum_i w(V_i)^\gamma \times pr(V_i) \tag{5}$$

The parameter γ ($\gamma \geqslant 1$) prevents lower priority vessels are systematically delayed. Note that this objective function is different to the classical tardiness concept in scheduling.

4.2. A meta-heuristic method for BAP and QCAP

We have developed three different methods for solving BAPs. Firstly, we applied the simplest solution, following the FCFS criteria: $\forall i$, $m(V_i) \leqslant m(V_{i+1})$. A vessel can be allocated at time t when there is no vessel moored in the berth or there are available contiguous quay length and QCs at time t (Algorithm 1).

Secondly, we also implemented a complete search algorithm to obtain the best (optimal) mooring order of vessels: the lowest T_w (lower bound of the function cost).

Finally, we developed a meta-heuristic GRASP algorithm for berth allocation (Algorithm 6). This is a randomly-biased multi-start method to obtain optimized solutions of hard combinatorial problems in a very efficient way.

Fig. 11 shows the application order of the different algorithms presented in this paper.

Algorithm 1: Allocating vessels following FCFS policy

```
Data: $V$: set of ordered incoming vessels;
Result: Sequence for $V$
$V_{last} \leftarrow \emptyset$;
$V_m \leftarrow \emptyset$;
foreach $V_i \in V$ do
    $t \leftarrow \max(m(V_{last}), a(V_i))$;
    $inst \leftarrow insertVessel(V_i, t)$;
    if $!inst$ then
        $T \leftarrow d(V_j) | V_j \in V_m \wedge d(V_j) > t$;
        while $t_k \in T \wedge !inst$ do
            $inst \leftarrow insertVessel(V_i, t_k)$;
        end
    end
    $V_{last} \leftarrow V_i$;
    $V_m \leftarrow V_m \cup V_i$;
end
```

Algorithm 1 (FCFS) uses the function *insertVessel* described in Algorithm 2 to allocate one vessel in a given time t (the required data are: V_i: Vessel for allocating; t: time for mooring the vessel; V_{in}: set of vessels already moored). If V_i can not moor at time t, it is repeated again each time one vessel (already moored) departs.

Algorithm 2: Function insertVessel. Allocating one vessel in the berth at time t

```
Data: $V_i$: Vessel for allocating; $t$: actual time; $V_{in}$: moored vessels;
Result: $V_i$ could moor
If $V_{in} = \varnothing$ then
    $m(V_i) \leftarrow a(V_i)$
    $q(V_i) \leftarrow \min\left(\texttt{maxQC}, \frac{l(V_i)}{\texttt{secQC}}\right)$;
    $d(V_i) \leftarrow m(V_i) + \frac{c(V_i)}{q(V_i) \times \texttt{movsQC}}$;
    return true;
else
    $q(V_i) \leftarrow \texttt{assignQC}(V_i, t, V_{in})$;
    if $q(V_i) = 0$ then
        return false;
    end
    $m(V_i) \leftarrow t$;
    $d(V_i) \leftarrow t + \frac{c(V_i)}{q(V_i) \times \texttt{movsQC}}$;
    $pos(V_i) \leftarrow \texttt{positionBerth}(V_i, V_{in})$;
    If $pos(V_i) < 0$ then
        return false;
    else
        return true;
    end
end
```

Algorithm 2 assigns the number of QCs ($q(V_i)$) to the vessel V_i through the function *assignQC* (Algorithm 3). This function takes into account the length of V_i, the security distance between two QCs (`secQC`) as well as the QCs used by the others moored vessels. Furthermore, Algorithm 2 obtains the berthing position ($pos(V_i)$) by the function *PositionBerth* (Algorithm 4). In order to get this position, we look for the first position where the vessel V_i fits according to its length ($l(V_i)$) and the security length between two contiguous vessels (`secLength`).).

Algorithm 3: Function assignQC. Determine how many QCs can be assigned to V_i

```
Data: $V_i$: Vessel for allocating; $t$: actual time; $V_{in}$: moored already vessels;
Result: Number of QCs
$cranes \leftarrow -1$; $cranes_m \leftarrow -1$;
repeat
    $nc \leftarrow \max(1, cranes)$;
    $t_f \leftarrow t + \frac{c(V_i)}{nc \times \texttt{movsQC}}$;
    $cranes_m \leftarrow nc$;
    $cranes \leftarrow \max\left(1, \min\left(\texttt{maxQC}, \texttt{floor}\left(\frac{l(V_i)}{\texttt{secQC}}\right)\right)\right)$;
    // Vessels which coincide with $V_i$
    $W \leftarrow v \in V_{in} | d(v) > t \wedge m(v) < t_f$;
    foreach $v \in W$ do
        // $QC_m$ is the number of used cranes when $v$ moors
        $QC_m \leftarrow \sum_{v' \in W} q(v') \mid m(v) \geqslant t \wedge m(v') \leqslant m(v) \wedge d(v') > m(v)$;
        // $QC_d$ is the number of used cranes when $v$ departs
        $QC_d \leftarrow \sum_{v' \in W} q(v') \mid d(v) \leqslant t_f \wedge m(v') < d(v) \wedge d(v') \geqslant d(v)$;
        $cranes \leftarrow \min(cranes, Q_C - QC_m)$;
        $cranes \leftarrow \min(cranes, Q_C - QC_d)$;
    end
    if $cranes \leqslant 0$ then
        return 0;
    end
until $cranes_m = cranes$;
return cranes;
```

In addition to the FCFS algorithm, the Complete Search was developed. The Complete Search uses the functions *moorVessel* (Algorithm 5) which in turn also uses the function *insertVessel* (Algorithm 2) to allocate one vessel from its arrival time. The required data for the Algorithm 5 is: v: Vessel for allocating; V_{in}: set of vessels already moored. However, with this search only a limited number of vessels can be taken into account since search space grows exponentially.

Algorithm 4: Function positionBerth. Determine the exact position of V_i at the berth

```
Data: $V_i$: Vessel for allocating; $V_{in}$: moored already vessels;
Result: Berthing position for $V_i$
// Set of vessels which coincide during the stay of $V_i$
$W' \leftarrow v \in V_{in} | d(v) \leqslant m(V_i) \wedge m(v) \geqslant d(V_i)$;
$W \leftarrow V_{in} - W'$;
sortByPositionBerth($W$);
$lastPos \leftarrow 0$; $lastLength \leftarrow 0$;
foreach $v \in W$ do
    // Security lengths
    $dLeft \leftarrow \max(lastLength, l(V_i)) \times \texttt{secLength}$;
    $dRight \leftarrow \max(l(v), l(V_i)) \times \texttt{secLength}$;
    $freeDist \leftarrow (lastPos + dLeft) - (pos(v) - dRight)$;
    if $freeDist \geqslant l(V_i)$ then
        // Assigning first free position
        return $lastPos + dLeft$;
    end
    $lastPos \leftarrow pos(v)$; $lastLength \leftarrow l(v)$;
end
// From the last vessel to the end of the berth
$dLeft \leftarrow \max(lastLength, l(V_i)) \times \texttt{secLength}$;
$freeDist \leftarrow (lastPos + dLeft) - L$;
if $freeDist \geqslant l(V_i)$ then
    return $lastPos + dLeft$;
end
return $-1$;
```

Therefore, we developed the meta-heuristic GRASP algorithm for berth allocation (Algorithm 6). Algorithm 6 is guided by the parameter δ $(0 \leqslant \delta \leqslant 1)$ which allows tuning of search randomization.

Algorithm 5: Function moorVessel. Allocating exactly one vessel in the berth

```
Data: $v$: vessel to moor; $V_{in}$: moored vessels;
$inst \leftarrow insertVessel(v, a(v))$;
if $!inst$ then
    $T \leftarrow d(v_j) | v_j \in V_{in} \wedge d(v_j) > t$;
    while $t_k \in T \wedge !inst$ do
        $inst \leftarrow insertVessel(v, t_k)$;
    end
end
```

Algorithm 6 receives as parameters both the δ factor and the set of vessels V_{out} waiting for mooring at the berth. Firstly, all the waiting vessels V_{out} are considered as candidates C. Each one of the candidate vessels is moored within the same state of the berth (Algorithm 5) assigning the mooring and departure times ($m(V_i), d(V_i)$), the number of QCs ($q(V_i)$) and the berthing position ($pos(V_i)$). These possibilities are evaluated according to the cost function f_c. This cost function is the sum of T_w that each vessel causes to the rest of unmoored vessels.

According to the cost function f_c, a restricted candidate list (RCL) is created. Then, one vessel v is chosen to be moored definitely following the random degree indicated by δ factor. Once v is determined, this is added to the set of vessels V_{in} and eliminated from the candidate list C. This loop is repeated until C is empty, it means, all the vessels are moored.

Algorithm 6: Allocating Vessels using GRASP metaheuristic

```
Data: $\delta$ factor; $V_{out}$ elements; $b$: state of the berth;
Result: $V_{in}$: set of scheduled vessels;
$V_{in} \leftarrow \{\}$;
$C \leftarrow V_{out}$;
while $C \neq \emptyset$ do
    foreach $v_e \in V_{out}$ do
        $f_c(v_e) \leftarrow 0$;
        $moorVessel(v_e, V_{in})$;
        $V'_{in} \leftarrow V_{in} \cup \{v_e\}$;
        foreach $v_o \in V_{out} | v_o \neq v_e$ do
            $moorVessel(v_o, V'_{in})$;
            $f_c(v_e) \leftarrow f_c(v_e) + (w(v_o) \times pr(v_o))$;
        end
    end
    $c_{inf} \leftarrow \min\{f_c(e) | e \in C\}$;
    $c_{sup} \leftarrow \max\{f_c(e) | e \in C\}$;
    $RCL \leftarrow \{e \in C | f_c(e) \leqslant c_{inf} + \delta(c_{sup} - c_{inf})\}$;
    $v \leftarrow random(RCL)$;
    $moorVessel(v, V_{in})$;
    $V_{in} \leftarrow V_{in} \cup \{v\}$; $C \leftarrow C - \{v\}$;
end
```

As metaheuristic GRASP indicates, this search is repeated according to the number of iterations specified by the user. Thus, the best solution according to T_w is returned as the solution for the BAP.

This metaheuristic process does not include a local search technique since it would involve testing the possible exchanges between the already ordered vessels, so that the computational cost would be increased considerably.

5. Evaluation

In this section, we evaluate the behavior of the algorithms developed in the paper. The experiments were performed on random instances. For the CStackP, containers are randomly distributed in blocks of 20 yard-bays, each one with six stacks of 4 tiers. A random instance of a yard-bay is characterized by the tuple $\langle n,s \rangle$, where 'n' is the number of containers and 's' ($s \leqslant n$) is the number of selected containers in the yard-bay. A random instance for the BAP has 'k' vessels with an arrival exponential distribution with vessel's data randomly fixed (lengths, drafts, moves and priorities).

In Table 1, we show the performance of our Planner H1 Sequential and H1 Ordered. These experiments were performed on blocks of containers composed by 10 yard-bays in the form $\langle 17,s \rangle$. Thereby, each yard-bay has a different number of goal containers. Additionally, we have established a timeout of 35 s to solve each yard-bay.

The first column in Table 1 corresponds to each instance solved by each planner. Thereby, row 1 corresponds to instance 1 of Planner H1 Ordered (1-O), row 2 corresponds to instance 1 of Planner H1 Sequential (1-S), and so on. The following 10 columns have two rows for each instance. They show for each instance the order in which the yard-bays are executed (upper row) and the number of goal containers that each one of them has (lower row). As example, for the instance 1-O, the sixth yard-bay is the first one in being executed and it has 9 goal containers. Finally, the last column presents the number of reshuffles needed to solve the instance or *Time Out* if a solution is not found.

As it can be observed, there are instances which only can be solved through the H1 Ordered, e.g. instances 1, 2 and 5. Moreover, other instances (instance 3), through the H1 Ordered planner, give a more efficient plan than the sequential one. However, there are also other examples in which both planners return the same plan (instance 4).

Table 1
Comparison of H1 Sequential and H1 Ordered.

Inst.	Order (Yard-Bay/N. Goal containers)										Number reshuffles
	1	2	3	4	5	6	7	8	9	10	
1-O	6	1	2	8	3	4	10	9	5	7	91
	9	7	6	6	5	5	5	4	2	2	
1-S	1	2	3	4	5	6	7	8	9	10	*Time Out*
	7	6	5	5	2	9	2	6	4	5	
2-O	2	5	6	1	4	3	8	9	10	7	64
	7	7	7	6	6	5	4	4	3	2	
2-S	1	2	3	4	5	6	7	8	9	10	*Time Out*
	6	7	5	6	7	7	2	4	4	3	
3-O	3	2	7	8	10	4	6	1	9	5	55
	8	6	6	6	6	5	5	4	4	1	
3-S	1	2	3	4	5	6	7	8	9	10	99
	4	6	8	5	1	5	6	6	4	6	
4-O	7	9	2	10	3	4	6	1	8	5	63
	10	7	6	6	5	5	4	3	3	2	
4-S	1	2	3	4	5	6	7	8	9	10	63
	3	6	5	5	2	4	10	3	7	6	
5-O	7	1	10	2	8	9	4	5	6	3	78
	9	7	7	6	6	6	5	5	4	2	
5-S	1	2	3	4	5	6	7	8	9	10	*Time Out*
	7	6	2	5	5	4	9	6	6	7	

Table 2
Values obtained through estimator *R*.

	it $P_2 = 15$		$P_2 = 17$		$P_2 = 19$	
	Reshuffles	*R*	Reshuffles	*R*	Reshuffles	*R*
$P_3 = 4$	2.8	2.6	3.7	4.6	4.6	4.8
$P_3 = 6$	4.2	4.8	6.2	5.9	5.6	6
$P_3 = 8$	6.2	6	8.6	8	7.2	8
Average values	4.4	4.5	6.2	6.2	5.8	6.3
St. deviation	2.1	1.9	3.1	2.1	2.6	1.9

Table 3
Computing time elapsed (seconds) for BAP.

No. vessels	5	10	11	12	13	15	20
Complete Search	<1	112	1105	11,830	57,462	–	–
Grasp	1	8	9	10	12	15	30

Table 4
Total waiting time elapsed.

Vessels	FCFS	CS
5 (separate arrival times)	73.72	46.10
10 (separate arrival times)	256.53	136.26
5 (close arrival times)	117.52	80.25
10 (close arrival times)	586.65	351.25

Thus, we can conclude that H1 Ordered can be considered a better planner than H1 Sequential to solve the complete block of yard-bays.

The actual average number of reshuffles and the average value of our estimator *R* are presented in Table 2. In each row, we present the average number of reshuffles form a set of 100 random instances. In all cases, we considered yard-bays with 4 tiers, so that the number of possible containers to be allocated to each yard-bay is set to 24 ($P_1 = 24$). The rest of parameters were increased: P_2 from 15 to 19 and P_3 from 4 to 8. It can be observed the similitude of the average number of reshuffles and *R* in all cases. It can be observed that the estimator *R* achieved values close to the actual values in all cases. The average value of *R* in all instances was very similar to the average value of the actual number of reshuffles. The standard deviation of *R* was even lower than the actual number of reshuffles due to the fact that it does not depend on the original allocation of out containers.

Table 3 shows the computational times (in seconds) required for solving BAP by using a complete search against the GRASP method with 1000 iterations. As observed, complete search is impracticable from 12 vessels (more than 3 h). However, the GRASP method takes around 30 s to solve a schedule of 20 vessels.

Table 4 shows the average waiting times using FCFS and Complete Search (CS) methods described for the BAP, with two different inter-arrival distributions (temporal separation among arriving vessels). Through these data, it is demonstrated that FCFS criteria results a schedule which is far away from the best one (CS). Besides, it can be observed that the denser the arrivals of the vessels are, the longer the waiting times are for such vessels. For example, given 10 vessels, the value of T_w is 136.26 with separate arrival times and it becomes 351.25 with close arrival times.

Using as minimization function the total weighted waiting time (T_w), Fig. 12 shows the results given by the FCFS criteria, and the GRASP procedure (with 1000 iterations) respect to the value of δ. The optimum value is $\delta = 0.3$, which indicates the suitability of the cost function used in the GRASP procedure (Algorithm 6). A total of 20 vessels are allocated, with two different inter-arrival distributions (separate and closest arrival times) among them.

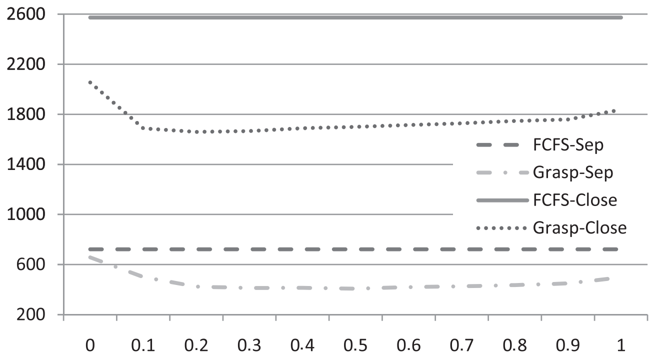


Fig. 12. Weighted waiting time (T_w) with FCFS and GRASP procedures.

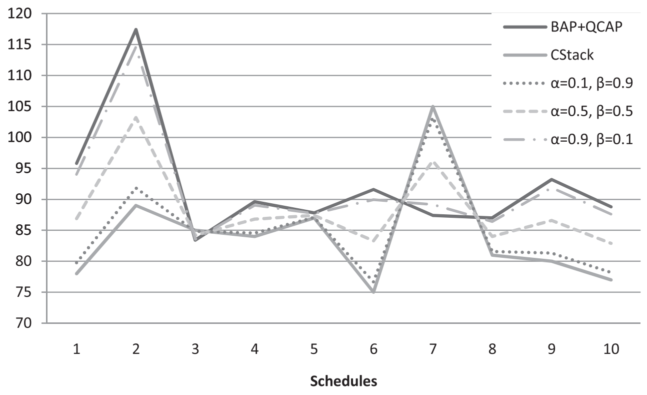


Fig. 13. Relating the costs of BAP + QCAP and CStackP.

As it was expected, the GRASP procedure obtains a lower T_w than the FCFS criteria. It is also remarkable that using GRASP is more profitable when the inter-arrival distribution of the vessels is closer. It is not possible to know the optimal T_w due to the exponential computational time required by a complete search with 20 vessels.

Finally, Fig. 13 shows the combined function cost Cost(Sol_i), introduced in Eq. (1) which relates for ten different scenarios:

- The normalized total weighted waiting time of vessels, Cost($SBAP_i$), and
- the number of its required container relocations, Cost($SCStackP_i$).

In each one of this ten cases, the arrival times and data of vessels, as well as the initial state of the container yard, have been randomly generated. Fig. 13 represents the combined function cost, Cost(Sol_i) with three different weights of the parameters α and β. We can see that better (or worst) berthing orders can require larger (or smaller) number of container relocations. For instance, with $\alpha = 0.5$, $\beta = 0.5$ the best choice is the sixth schedule. It does not get the best solution for BAP + QCAP, however it corresponds to the schedule with the smallest number of container relocations (CStackP).

6. Conclusions

The Container Stacking Problem, the berth allocation problem and the Quay Crane Assignment Problem are three important and related problems in maritime container terminals. In this paper, we have presented a decision support system to manage these problems in a coordinated way. To this end, we have developed a domain-oriented heuristic planner for calculating the number of reshuffles needed to allocate the containers in the appropriate place. Furthermore, we have estimated this number in order to avoid the planning phase in unnecessary instances. Then, we have developed a GRASP metaheuristic for the berth allocation problem and the Quay Crane Assignment Problem which generates an optimized order of vessels to be served according to existing berth constraints. By combining these optimized solutions in our integrated system, terminal operators can be assisted to decide the most appropriated solution in each particular case.

As future work, we are working on improving the function to estimate the number of reshuffles by adding some more specific parameters. Furthermore we plan to improve the GRASP method and to adequate the parameters (α, β and γ) to practical decisions and expert knowledge. Then, the developed system, as a computer-based aid system, could assist container terminal's operator to simulate, evaluate and compare different feasible alternatives.

Acknowledgment

This work has been partially supported by the research projects TIN2010-20976-C02-01 (Min. de Educación y Ciencia, Spain-FEDER), P19/08 (Min. de Fomento, Spain-FEDER), and the VALi+d Program of the Conselleria d'Educació (Generalitat Valenciana), as well as with the collaboration of the maritime container terminal MSC (Mediterranean Shipping Company S.A.).

References

[1] Global container terminal operators annual review and forecast, Annual Report, 2010.
[2] M. Aleb-Ibrahimi, B. de Castilho, C.F. Daganzo, Storage space vs handlingwork in container terminals, Transportation Research-B 27B (1993) 13–32.
[3] C. Bierwirth, F. Meisel, A survey of berth allocation and quay crane scheduling problems in container terminals, European Journal of Operational Research 202 (2010) 615–627.
[4] A. Bruzzone, R. Signorile, Simulation and genetic algorithms for ship planning and shipyard layout, Simulation 71 (1998) 74–83.
[5] S.H. Chen, J.N. Chen, Forecasting container throughputs at ports using genetic programming, Expert Systems with Applications 37 (3) (2010) 2054–2058.
[6] C.Y. Cheong, K.C. Tan, D.K. Liu, Solving the berth allocation problem with service priority via multi-objective optimization, in: IEEE Symposium on Computational Intelligence in Scheduling, 2009, CI-Sched'09, 2, 2009 March to 30, 2009, pp. 95–102.
[7] T.N. Chuang, C.T. Lin, J.Y. Kung, M.D. Lin, Planning the route of container ships: a fuzzy genetic approach, Expert Systems with Applications 37 (4) (2010) 2948–2956.
[8] J.F. Cordeau, G. Laporte, P. Legato, L. Moccia, Models and tabu search heuristics for the berth-allocation problem, Transportation Science 39 (4) (2005) 526–538.
[9] T.A. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures, Journal of Global Optimization 6 (2) (1995) 109–133.
[10] L.M. Gambardella, A.E. Rizzoli, M. Zaffalon, Simulation and planning of an intermodal container terminal, Simulation 71 (1998) 107–116.
[11] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL – the planning domain definition language, AIPS-98 Planning Committee (1998).
[12] G. Giallombardo, L. Moccia, M. Salani, I. Vacca, Modeling and solving the tactical berth allocation problem, Transportation Research Part B: Methodological 44 (2) (2010) 232–245.
[13] Y. Guan, R.K. Cheung, The berth allocation problem: models and solution methods, OR Spectrum 26 (1) (2004) 75–92.
[14] C. Gutiérrez, I. García-Magariño, Modular design of a hybrid genetic algorithm for a flexible job-shop scheduling problem, Knowledge-Based Systems 24 (1) (2011) 102–112.
[15] M. Han, P. Li, J. Sun, The algorithm for berth scheduling problem by the hybrid optimization strategy GASA, in: Proceedings of the Ninth International Conference on Control, Automation, Robotics and Vision (ICARCV 2006), 2006, pp. 1–4.
[16] P. Hansen, C. Ceyda Oguz, N. Mladenovic, Variable neighborhood search for minimum cost berth allocation, European Journal of Operational Research 191 (2008) 636–649.
[17] L. Henesey, overview of transshipment operations and simulation, in: MedTrade Conference, Malta, April 2006, pp. 6–7.
[18] Jörg Hoffmann, The metric-ff planning system: translating ignoring delete lists to numeric state variables, Journal of Artificial Intelligence Research 20 (1) (2003) 291–341.
[19] A. Imai, K.I. Nagaiwa, Chan Weng Tat, Efficient planning of berth allocation for container terminals in Asia, Journal of Advanced Transportation 31 (1) (1997) 75–94.
[20] K.H. Kim, G.P. Hong, A heuristic rule for relocating blocks, Computers and Operations Research 33 (4) (2006) 940–954.
[21] K.K. Lai, K. Shih, A study of container berth allocation, Journal of Advanced Transportation 26 (1) (1992) 45–60.
[22] Yusin Lee, Nai-Yun Hsu, An optimization model for the container pre-marshalling problem, Computers and Operations Research 34 (11) (2007) 3295–3313.
[23] G. Mauri, A. Oliveira, L. Lorena, A hybrid column generation approach for the berth allocation problem, in: Eighth European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2008), vol. 4972, 2008, pp. 110–122.
[24] R. Micalizio, P. Torasso, On-line monitoring of plan execution: a distributed approach, Knowledge-Based Systems 20 (2) (2007) 134–142.
[25] B. Naderi, R. Tavakkoli-Moghaddam, M. Khalili, Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan, Knowledge-Based Systems 23 (2) (2010) 77–85.
[26] E. Nishimura, A. Imai, S. Papadimitriou, Berth allocation planning in the public berth system by genetic algorithms, European Journal of Operational Research 131 (2001) 282–292.
[27] K. Park, T. Park, K.R. Ryu, Planning for remarshaling in an automated container terminal using cooperative coevolutionary algorithms, in: Proceedings of the 2009 ACM Symposium on Applied Computing, ACM, 2009, pp. 1098–1105.
[28] M. Salido, O. Sapena, F. Barber, The container stacking problem: an artificial intelligence planning-based approach, in: The International Workshop on Harbour, Maritime and Multimodal Logistics Modelling and Simulation, 2009.
[29] M. Salido, O. Sapena, M. Rodriguez, F. Barber, A planning tool for minimizing reshuffles in containers terminals, in: ICTAI 2009: 21st International Conference on Tools with Artificial Intelligence, 2009.
[30] R. Stahlbock, S. Voß, Operations research at container terminals: a literature update, OR Spectrum 30 (1) (2008) 1–52.
[31] S. Theofanis, M. Boile, M.M. Golias, Container terminal berth planning, Transportation Research Record: Journal of the Transportation Research Board 2100 (1) (2009) 22–28.
[32] I.F.A. Vis, R. De Koster, Transshipment of containers at a container terminal: an overview, European Journal of Operational Research 147 (2003) 1–16.
[33] Terry Winograd, Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, MIT. Cent. Space Res., Cambridge, MA, 1971.
[34] P. Zhou, H. Kang, L. Lin, A dynamic berth allocation model based on stochastic consideration, in: Proceedings of the Sixth World Congress on Intelligent Control and Automation (WCICA 2006), 2006.