Appl Intell (2014) 40:273–290
DOI 10.1007/s10489-013-0462-4

A GRASP-based metaheuristic for the Berth Allocation Problem and the Quay Crane Assignment Problem by managing vessel cargo holds

Mario Rodriguez-Molins · Miguel A. Salido · Federico Barber

Published online: 13 August 2013

Abstract Container terminals are open systems that generally serve as a transshipment zone between vessels and land vehicles. These terminals carry out a large number of planning and scheduling tasks. In this paper, we consider the problem of scheduling a number of incoming vessels by assigning a berthing position, a berthing time, and a number of Quay Cranes to each vessel. This problem is known as the Berth Allocation Problem and the Quay Crane Assignment Problem. Holds of vessels are also managed in order to obtain a more realistic approach. Our aim is to minimize the total waiting time elapsed to serve all these vessels. In this paper, we deal with the above problems and propose an innovative metaheuristic approach. The results are compared against other allocation methods.

Keywords Planning · Scheduling · Optimization methods · Algorithms · Metaheuristic · GRASP

1 Introduction

Container terminals generally serve as a transshipment zone between ships and land vehicles (trains or trucks). In [12], it is shown how this transshipment market has grown rapidly. Between 1990 and 2008, container traffic grew from 28.7 million TEU (Twenty-foot Equivalent Unit) to 152.0 million TEU, an increase of about 430 %. This corresponds to an average annual compound growth of 9.5 %. In the same period, container throughput went from 88 million TEU to 530 million TEU, an increase of 500 %, which is equivalent to an average annual compound growth of 10.5 %. The surge of both container traffic and throughput is linked with the growth of international trade as well as to the adoption of containerization as the privileged vector for maritime shipping and inland transportation [5]. However, the Global Economic Crisis of 2008 has had a negative impact over the container traffic [25].

The evolution of the container traffic has motivated the artificial intelligence research community to develop optimization techniques to manage combinatorial problems related to seaport terminals efficiently. These problems occur in transportation [4, 34] as well as within the container terminals. Container terminals are open systems where terminal operators must face combinatorial optimization problems to ensure the fast loading and unloading of the vessels. In container terminals, there are three distinguishable areas (see Fig. 1): the berth area, where vessels are berthed for service; the storage yard, where containers are temporarily stored to be exported or imported; and the terminal receipt and delivery gate area, which connects the container terminal to the hinterland. Each one presents different planning and scheduling problems to be optimized [15, 32, 33]. For example, berth allocation, quay crane assignment, stowage planning, and quay crane scheduling must be managed in the berthing area; the container stacking problem, yard crane scheduling, and horizontal transport operations must be carried out in the yard area; and the hinterland operations must be solved in the land side area. Figure 2 shows the main planning and scheduling problems that must be managed in a container terminal [37].

M. Rodriguez-Molins (✉) · M.A. Salido · F. Barber
Instituto de Automática e Informática Industrial, Universitat Politècnica de València (UPV), Camino de vera, s/n, 46022, Valencia, Spain
e-mail: mrodriguez@dsic.upv.es

M.A. Salido
e-mail: msalido@dsic.upv.es

F. Barber
e-mail: fbarber@dsic.upv.es

We focus our attention on the Berth Allocation Problem (BAP) and the Quay Crane Assignment Problem (QCAP) taking into account the holds of each vessel. The former is a well-known, NP-hard combinatorial optimization problem [22], which consists of assigning incoming vessels to berthing positions. The latter deals with assigning a certain number of QCs to each vessel such that all required movements of containers can be fulfilled.

Containers to be loaded/unloaded in the vessel are stored on the deck as well as in the holds. The holds of the vessels are structures that speed up both loading and unloading and keep the containers secure while at sea. Once a vessel arrives at the port, it waits at the roadstead until it has permission to moor at the quay. The quay is a platform that protrudes 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 (QCs), that are 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 transportation mode (e.g., rail or road).

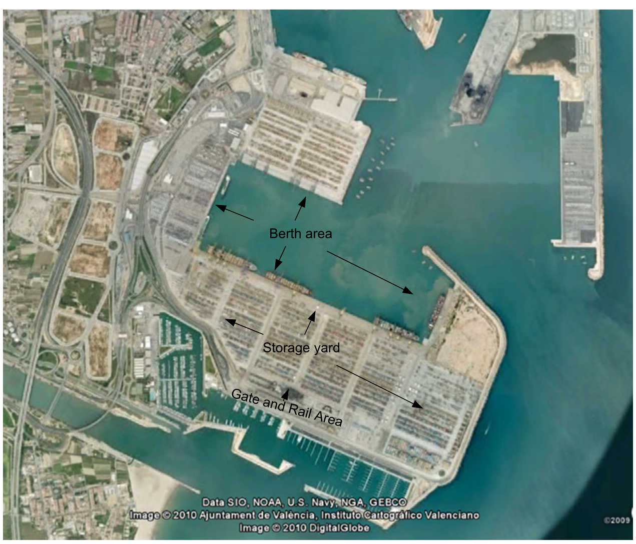
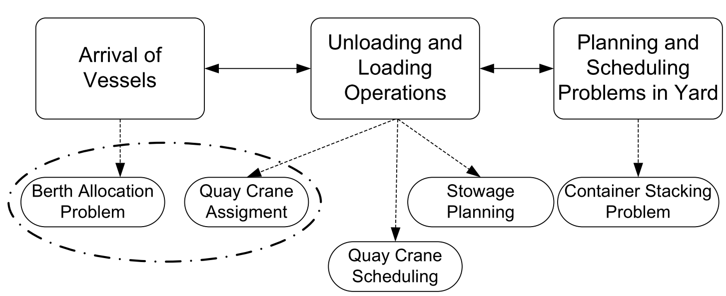


Fig. 1 Container terminal in Valencia

Fig. 2 Planning and scheduling problems in container terminals



Managers at container terminals are confronted with two interrelated decisions: *where* and *when* the vessels should moor. First, they have to take into account physical restrictions such as length or draft, and they also have priorities to take into account, as well as other aspects to minimize both port and user costs, which are usually opposites. Generally, this process is solved manually. It is usually solved by means of a policy to serve the first vessel that arrives (FCFS). Figure 3 shows an example of the graphical space-time representation of a berth plan with 6 vessels. Each rectangle represents a vessel with its handling time and length. For instance, vessel 4 must moor after vessels 1 and 2 depart.

The overall collaboration goal of our group at the Universitat Politècnica de València (UPV), the 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 allocating spaces to outbound containers [28, 29], to berth incoming vessels [31], to identify bottlenecks, to determine the consequences of changes [30], to provide support in the resolution of incidences, to provide alternative berthing plans, etc. Researchers commonly develop mathematical or statistical models that represent real-world systems. Nevertheless, these systems are very complex and composed of different problems that sometimes have opposing goals. These problems must be simplified with several assumptions in order to

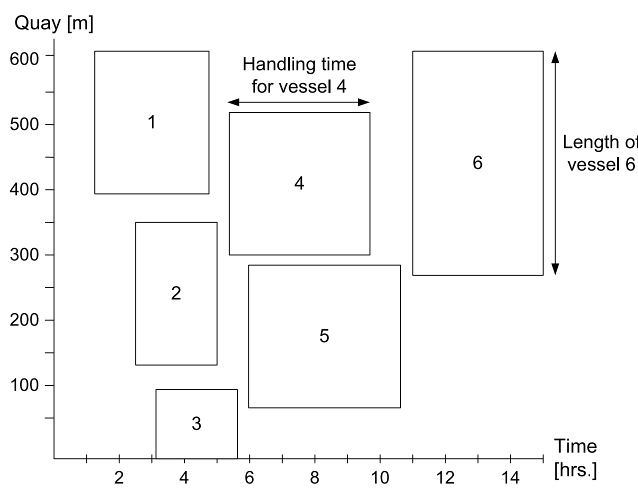


Fig. 3 A berth plan with 6 vessels

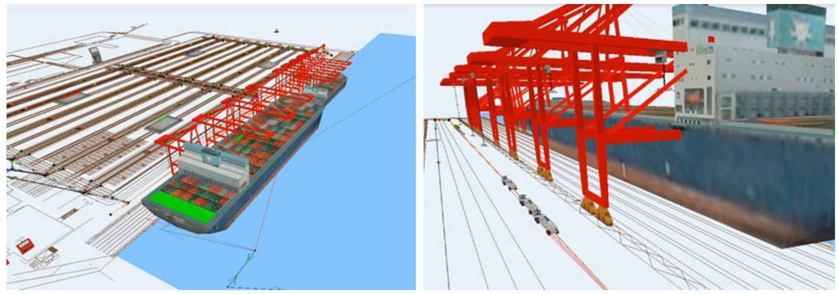


Fig. 4 Simulator developed within the Masport Project [36]

Fig. 5 A classification scheme for BAP formulation [2]

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

be appropriately modeled. Mathematical models are necessary but exact optimization methods algorithms cannot obtain the optimal solution in a reasonable time. Thus, techniques from the artificial intelligence field (such as local search or metaheuristics) must be applied in order to solve these combinatorial optimization problems efficiently and get near-optimal solutions in an efficient way [1]. Our artificial intelligence techniques are included within a simulator that is able to represent a given state of the terminal and simulate the behavior in different parts of the terminal (see Fig. 4). In this paper, a metaheuristic has been developed to schedule the incoming vessels and has been compared with several real world rules, commonly used by expert terminal operators.

The remainder of this paper is organized as follows. Section 2 presents a review of the literature about the BAP and QCAP and different techniques to manage them. Section 3 presents the problem definition. Section 4 explains the whole process of mooring one vessel and Sect. 5 describes the developed metaheuristic technique. Section 6 presents the computational results, and Sect. 7 summarizes our conclusions.

2 Literature review

In [35], the authors present 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 operators and shipping lines.

To show similarities and differences in the existing models for berth allocation, a classification scheme is presented in [2] (see Fig. 5). 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 that vessel handling times are considered in the problem. The fourth attribute defines the performance measure to reflect different service quality criteria. The most important are the criteria that minimize the waiting time (*wait*) and the handling time (*hand*) of a vessel. Both these 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. Thus, by using the above classification scheme, a certain type of BAP is described by a selection of values for each one of the attributes. For instance, given a problem where the quay is assumed to be a continuous line (*cont*). The arrival times restrict the earliest berthing of vessels (*dyn*), and the handling times depend on the berthing position of the vessel (*pos*). The objective is to minimize the sum of the waiting time (*wait*) and the handling time (*hand*). According to the scheme proposed by [2], this problem is classified by $cont|dyn|pos|\Sigma(wait + hand)$.

One of the early works that appeared in the literature developed a heuristic algorithm by considering a First-Come-First-Served (FCFS) rule [17]. However, some authors maintain the idea that for high port throughput, optimal vessel-to-berth assignments should be found without considering the FCFS policy [13]. Therefore, in this paper, we use the FCFS policy in order to get an upper bound. Nevertheless, this approach may result in some vessels' dissatisfaction regarding the order of service.

Several heuristic and metaheuristic approaches have been developed to solve different problems in container terminals. In [2], the authors give a 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 that are distributed among discrete problems, continuous problems, and hybrid problems are presented. These problems have been mostly considered separately and with an interest mainly focused on BAP [3, 6, 11].

In the integration of both problems, BAP+QCAP, different approaches have been developed considering a discrete quay line, specially genetic algorithms. In [14], a solution based on genetic algorithms is presented for the integration of BAP with the QCAP with the objective of minimizing the total service time. A hybrid genetic algorithm is also presented in [21] where they minimize the sum of the handling time, the waiting time, and the delay time for every ship. In this sense, [10] presents the integration through two mixed integer programming formulations including a tabu search method (adapted from [6]), with the objective of minimizing the yard-related housekeeping costs that are generated by the flows of containers exchanged between vessels.

The integration of BAP+QCAP when the quay line is continuous was first introduced in [26] with a method of two phases. In the first phase, a Lagrangian relaxation based heuristic is used to obtain the berthing position and the number of QCs, and the second phase applies a dynamic programming to obtain the detailed schedule of the QCs. In [24], two different heuristics are presented to solve this model: squeaky wheel optimization and tabu search that show significant results compared with solutions reported by [26].

Focusing only on the holds of one vessel, a genetic algorithm able to solve the quay crane scheduling problem is presented in [19] by determining a handling sequence of holds for the quay cranes assigned to a vessel. Furthermore, in [19], the NP-completeness of this problem for one vessel was proved. The integration of BAP+QCAP problems considering the holds of vessels was first studied by [7] and [27], but they did not consider the interference among QCs. There are similar problems solved by MILP approaches [23]. Results obtained by these exact approaches show that this problems needs to be decomposed into two phases and they cannot solve realistic problems of medium size in a reasonable time.

Our approach deals with the integration of these two problems (BAP+QCAP) through a metaheuristic called Greedy Randomized Adaptive Search Procedures (GRASP) [8], taking into account the requirements of container operators of MSC (Mediterranean Shipping Company S.A.). This metaheuristic is able to find optimized solutions within an acceptable computation time in a very efficient way and it has been applied in a wide range of combinatorial optimization problems [9]. Moreover, several GRASP-based approaches have been developed and applied in different container terminal problems [16, 20].

Following the classification scheme in [2], our approach is represented by *BAP*, *QCAP*; and specifically, the BAP is defined as by $cont|dyn|QCAP|\Sigma wait$. Thus, we focused on the following attributes and performance measure:

- *Spatial attribute*: we assume that the quay is a continuous line (*cont*), so there is no partitioning of the quay and the 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, but it utilizes quay spaces better [2].
- *Temporal attribute*: we assume dynamic problems (*dyn*) where arrival times restrict the earliest berthing times. Since fixed arrival times are given for the vessels, vessels cannot berth before their expected arrival time.
- *Handling time attribute*: we assume that the handling time of a vessel depends on the assignment of QCs (*QCAP*).
- *Performance measure*: Our objective is to minimize the sum of the waiting time (*wait*) of all the scheduled vessels to be served.

This paper presents two approaches for modelling QCAP:

– *Static*: QCs are assigned to one vessel i and they cannot move to another vessel j until the vessel i leaves the container terminal.
– *Dynamic*: QCs are assigned to the holds of the vessel. Thus, once all the movements of one hold are done, the QC can move to another location (another hold in the same or other vessel).

Our approach presents a dynamic and continuous berthing model that takes into account the QCs and the holds of the vessels in order to obtain the handling time. Most of the studies presented above are focused on discrete models without managing the holds of the vessels. Unlike the models presented above having regard to the holds, our approach manages the constraints related to the cranes. Furthermore, our approach differs from crane scheduling problems in that several vessels with different arrival times are the input data.

In the rest of the paper, we specify the above problems (BAP+QCAP, including management of holds) and propose an innovative GRASP-based metaheuristic approach. The results obtained with several scenarios are compared to other allocation methods, contrasting the usefulness of our proposal by efficiently obtaining optimized solutions to these problems.

3 Problem description

The objective in BAP+QCAP is to obtain an optimal distribution of the docks and cranes for vessels waiting to berth. An optimal distribution that takes into account specific constraints (length and depth of vessels, ensuring a correct order for vessels that exchange containers, ensuring departure times, etc.) and optimization criteria (priorities, minimization of waiting and staying times of vessels, satisfaction with the order of berthing, minimization of crane moves, degree of deviation from a pre-determined service priority, etc.). When the quay is discrete (it is divided in berths), the BAP could be considered as a special kind of machine scheduling problem, where the job and machine are the vessel and the berth, respectively. In machine scheduling, only the starting times of jobs are determined, but in continuous BAPs the berthing positions are also necessary for the output schedule.

In the following, we introduce the notation used for each vessel:

V The set of incoming vessels. Each vessel is denoted as $V_i \in V$.

QC Available QCs in the container terminal. These QCs are identical in terms of the productivity of loading/unloading containers. The parameters of QCs are:

- `movsQC` Number of QC moves per time unit.
- $\mathtt{HH}_{\mathtt{QC}}$ Time units required to reallocate the QC to another hold from the same vessel (Hold-to-Hold movement).
- $\mathtt{VV}_{\mathtt{QC}}$ Time units required to move the QC to another vessel (Vessel-to-Vessel movement).

L Total length of the berth in the container terminal.

$a(V_i)$ Arrival time of the vessel V_i at port.

$l(V_i)$ Length of V_i. There is a safe distance (`safeLength`) between two moored vessels: we assume 5 % of the length of the vessels.

$pr(V_i)$ Vessel priority.

$h(V_i)$ Number of holds in V_i. All the holds of the vessels have the same width.

$c_j(V_i)$ Number of required movements to load or unload containers from/into the hold j, $1 \le j \le h(V_i)$. The handling time of each hold j $(1 \le j \le h(V_i))$ is given by: $\frac{c_j(V_i)}{\mathtt{movsQC}}$.

$m(V_i)$ Mooring time of V_i.

$p(V_i)$ Berthing position where V_i will moor.

$q(V_i)$ Number of assigned QCs to V_i. The maximum number of assigned QCs by vessel depends on its length since a safety distance is required between two contiguous QCs (`safeQC`) and the maximum number of QCs that the container terminal allows per vessel (`maxQC`).

$st_j(V_i)$ Starting time of QC j at Vessel V_i, $1 \le j \le q(V_i)$. Only one QC can be assigned to one hold.

$ht_j(V_i)$ Handling time of the QC j, $1 \le j \le q(V_i)$.

$h_j(V_i)$ Set of handling times of each hold assigned to the QC j, $1 \le j \le q(V_i)$.

$d(V_i)$ 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 its arrival at port until it moors: $w(V_i) = m(V_i) - a(V_i)$.

To simplify the problem, we assume that mooring and unmooring does not consume time, simultaneous berthing is allowed, and every vessel has a draft lower than or equal to the water-depth of the berth. Furthermore, once a QC starts to work in a hold, it must complete it without any pause or shift (non-preemptive tasks). When a QC finishes the movements of one hold, it can move to another hold from the same vessel or to another vessel.

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

$$T_w = \sum_{V_i \in V} \left[\left[w(V_i) \right]^{\gamma} \times pr(V_i) \right] \quad (1)$$

The parameter γ ($\gamma \ge 1$) prevents lower priority vessels from being systematically delayed. Thus, the component of

Table 1 Example of the γ parameter over two different schedules

Schedule	$pr(V_1)$	$pr(V_2)$	$w(V_1)$	$w(V_2)$	T_w ($\gamma = 1$)	T_w ($\gamma = 1.2$)
A	6	2	150	500	1900	5917
B	6	2	220	310	1940	5835

each vessel in the optimization function is not exactly linear with its waiting time ($w(V_i)$). In this way, vessels with large waiting times ($w(V_i)$) will have a proper weighting in the objective function, although they have low priority values ($p(V_i)$). For instance, let A and B be two alternative schedules for two vessels V_1 and V_2 with $pr(V_1) = 6$ and $pr(V_2) = 2$, respectively (see Table 1). In the schedule A, the waiting time of V_2 ($w(V_2) = 500$) is much larger than V_1 ($w(V_1) = 150$); whereas the waiting times in schedule B are closer to each other ($w(V_1) = 220$; $w(V_2) = 310$). Table 1 also shows the values of the objective functions with respect to the γ value for both schedules. If γ is not considered in the objective function ($\gamma = 1$), it is preferable to chose the schedule A ($T_w = 1900$) although V_2 must wait 500 time units. However, with a γ value greater than 1 ($\gamma = 1.2$), the schedule B, which balances the waiting times of V_1 and V_2, is chosen as the best schedule ($T_w = 5835$) avoiding that the vessel with the lowest priority (V_2) is delayed.

There are two other key factors within container terminals: the ratio of berth usage (B_u), and the quay crane throughput (T_{qc}). Berth usage is obtained by Eq. (4). It reflects the area held by vessels with respect to the maximum area. The maximum area depends on the length of the quay (L) and the mooring time of the first vessel calculated by (2) and the departure time of the last vessel calculated by (3).

$$firstArrival = \min_{V_i \in V} \{m(V_i)\} \tag{2}$$

$$lastDeparture = \max_{V_i \in V} \{d(V_i)\} \tag{3}$$

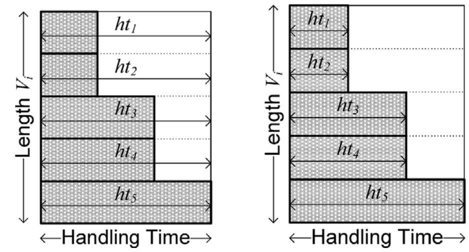
Thus, the ratio of the berth usage (B_u) is:

$$B_u = \frac{\sum_{V_i \in V} [l(V_i) \times (d(V_i) - m(V_i))]}{L \times (lastDeparture - firstArrival)} \tag{4}$$

The QC throughput factor (T_{qc}) depends on the model for the QCAP under consideration. The static QCAP model calculates T_{qc} by means of (5), taking into consideration that one QC remains at the same vessel until it departs. Thereby, all the QCs are assigned to vessel V_i up to its departure time $d(V_i)$ even though these QCs are not moving any container from/to the vessel:

$$\text{Static QCAP: } T_{qc} = \sum_{V_i \in V} \left[q(V_i) \times \left(d(V_i) - m(V_i)\right)\right] \tag{5}$$

$$\text{Dynamic QCAP: } T_{qc} = \sum_{V_i \in V} \sum_{1 \le j \le q(V_i)} ht_j(V_i) \tag{6}$$



(a) Static (b) Dynamic

Fig. 6 Differences between static and dynamic models for calculating T_{qc} for QCAP

However, the dynamic QCAP model uses (6). This model considers that once one QC finishes its task at one hold, it can move to another location. Therefore, the time that each QC is busy is just its handling time.

Figure 6 shows an example of a vessel containing 5 holds with a different number of containers each, thus with different handling times for each hold. In this example, a QC has been allocated to each hold. The shaded rectangle indicates the time a QC works on a hold and the arrow represents the time that the QC is assigned to the vessel. In the static QCAP model (see Fig. 6(a)), they stay until the vessel departs, while the dynamic QCAP model allows to move one QC to another hold or vessel before the vessel departs (see Fig. 6(b)).

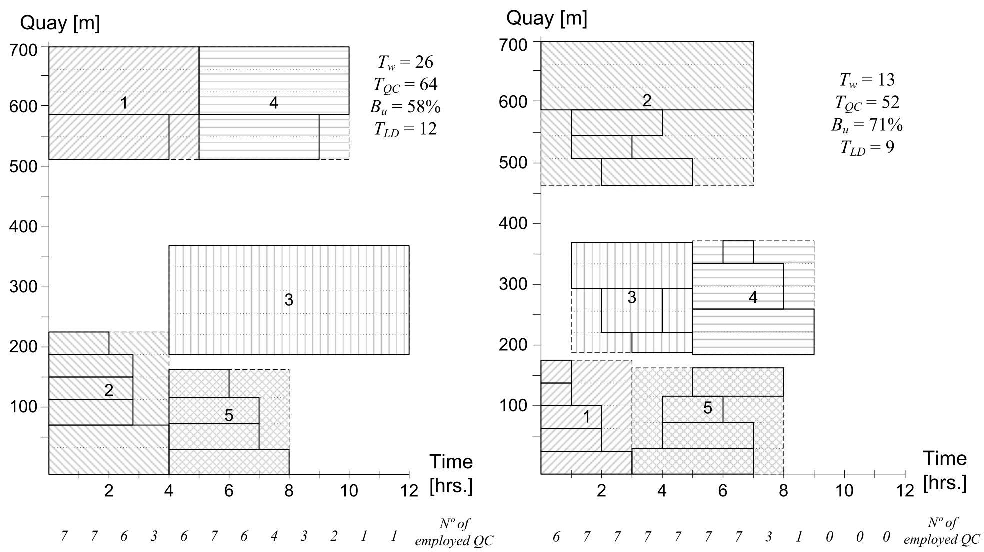
By taking into consideration the holds ($h(V_i)$) of each vessel, our model makes better use of the resources (QCs and berth) as shown in Fig. 7. In this figure, a schedule of 5 vessels is shown. Each dashed rectangle represents a vessel with its id number and each bold rectangle represents the time that one QC is working on a hold.

Figure 7(a) shows the static QCAP model. Thus, when one QC is assigned to one vessel i, this QC cannot be moved to another vessel j until vessel i leaves the container terminal. Figure 7(b) shows the dynamic QCAP model and introduces the concept of holds. Once a QC finishes a task that is related to one hold in the vessel i, it could keep working on another hold of the same vessel or move to another vessel j. Thus, the dynamic model obtains the following: a departure time of the last vessel (T_{LD}) that is earlier than the static model (from 12 to 9 time units); a waiting time (T_w) that is lower than the static one (from 26 to 13 time units). Thus, we can see that the berth usage ratio is increased by 20 %. In other words, with dynamic QCAP model more QCs are used per time unit than with static QCAP, and the total time that the QCs are tied to one vessel (T_{qc}) is also reduced.

4 Mooring one vessel

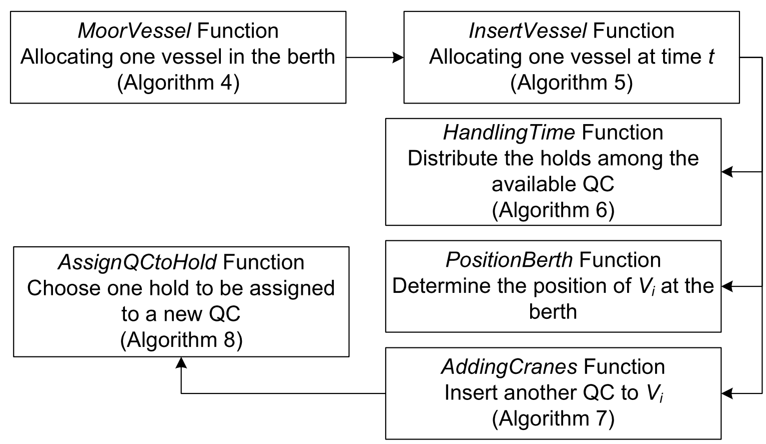
Once the problem has been defined, the whole process to moor one vessel is presented. The *MoorVessel* function (Algorithm 1) moors one vessel V_i as early as possible, starting

Fig. 7 Plans obtained with and without the holds



(a) QCs assigned to one vessel (static) (b) QCs assigned to holds of one vessel (dynamic)

Fig. 8 Execution order of the presented algorithms



at its arrival time ($a(V_i)$). This function checks whether V_i can moor at time t or when a QC has completed its task (steps 11 to 18). The data required for this function is the vessel to allocate resources (V_i) and the set of moored vessels scheduled previously by the algorithm (V_m). There are three steps in this mooring process (see Fig. 8):

1. To verify whether there are QCs available during the handling time of V_i (Algorithm 2).
2. To make sure there is enough continuous length at the berth to moor V_i.
3. To assign more QCs when it is possible (Algorithm 4).

The *InsertVessel* function (Algorithm 2) performs, if it is possible, the steps needed to allocate one vessel at the given time t. First, a quick check of the available quay length is carried out in order to know if there would be enough space to moor V_i (steps 2 to 5). Then, the maximum number of QCs for V_i is calculated from its mooring time until its departure (steps 6 to 34). This process starts assigning just one QC and increases this number according to the available QCs found during the vessel handling time (t, t_f). Available QCs are obtained by counting the number of cranes used by the other vessels (*CranesWorking* function) at their mooring time and whenever one of their assigned QCs has completed its task. As we have mentioned above, in this paper, we consider that each QC is assigned to just one hold; therefore initially, the holds of one vessel ($h(V_i)$) are distributed among the different QCs by the *HandlingTime* function (Algorithm 3). Finally, the berthing position is calculated by the *PositionBerth* function, and, when it is possible, additional QCs are assigned to V_i by the *AddingQuayCranes* function (Algorithm 4).

The allocation of the holds of V_i to the different QCs is done in two phases (Algorithm 3). Step 2 calculates the time needed to complete all the movements of each hold. Later, in steps 6–17, each hold is assigned to the given QCs in descending order according to their handling times.

Algorithm 1 `MoorVessel` function. Allocating exactly one vessel in the berth

Input: V_i: vessel to moor; V_m: Vessels already moored;
Output: V_i: vessel with all the resources allocated;

if $|V_m| = 0$ **then**
  // There is no other moored vessel
  $nQC := \max(1, \min(\mathtt{maxQC}, \lfloor \frac{l(v)}{\mathtt{safeQC}} \rfloor))$;
  HandlingTime(V_i, nQC);
  $m(V_i) := a(V_i)$;
  $d(V_i) := m(V_i) + \max_{1 \le j \le q(V_i)}(st_j(V_i) + ht_j(V_i))$;
  $q(V_i) := nQC$;
  $p(V_i) := 0$;
else
  // There are other vessels. Moor at the earliest possible time
  $T \leftarrow \{a(V_i)\} \cup \{(st_k(j) + ht_k(j)) \mid j \in V_m, 1 \le k \le q(j) \wedge \; st_k(j) + ht_k(j) > a(V_i)\}$;
  Sort(T); // Sort in ascending order
  for all $t_k \in T$ **do**
    $inst$:=InsertVessel$(V_i, t_k + \mathtt{VV}_{\mathtt{QC}}, V_m)$;
    if $inst$ **then**
      break
    end if
  end for
end if
return V_i;

Algorithm 2 `InsertVessel` function. Allocating one vessel in the berth at time t

Input: V_i: Vessel to allocate; t: Actual time; V_m: Vessels already moored;
Output: A boolean indicating whether V_i could moor or not;

// Check the length available at the quay
$L_{avail} \leftarrow L - (\sum_{v_j \in V_m} l(v_j) \mid m(v_j) \le t \wedge d(v_j) > t)$;
if $L_{avail} \le l(v)$ **then**
  return False;
end if

$cranes := -1$; $cranes_m := -1$;
repeat
  $nc := \max(1, cranes)$;
  HandlingTime(V_i, nc); // Handling time given the nc QCs
  $t_f := t + \max(st_j(V_i) + ht_j(V_i)),\ 1 \le j \le q(V_i)$;

  // Vessels which coincide with V_i
  $W \leftarrow \{v \in V_m \mid d(v) > t \wedge m(v) < t_f\}$;

  // Find the maximum number of QCs available to be used in the interval (t, t_f)
  $cranes_m := nc$
  $cranes := \max(1, \min(\mathtt{maxQC}, \lfloor \frac{l(V_i)}{\mathtt{safeQC}} \rfloor))$;
  $QC_u := \sum_{\forall v \in W}$ CranesWorking(v, t);
  $cranes := \min(cranes, QC - QC_u)$;
  for all $i \in W$ **do**
    if $m(i) \ge t$ **then**
      $QC_u := \sum_{\forall v \in W}$ CranesWorking$(v, m(i))$;
      $cranes := \min(cranes, QC - QC_u)$;
    end if
    for $j \leftarrow 1$ **to** $q(i)$ **do**
      $t_q := st_j(v) + ht_j(v)$;
      if $t_q \ge m(V_i) \wedge t_q < d(V_i)$ **then**
        $QC_u := \sum_{\forall v \in W}$ CranesWorking(v, t_q);
        $cranes := \min(cranes, QC - QC_u)$;
      end if
    end for
  end for
  if $cranes \le 0$ **then**
    return False;
  end if
until $cranes_m \le cranes$; // Repeat until the number of available QCs does not change

// Assign the number of QCs and the mooring and departure times
$q(V_i) := cranes_m$;
$m(V_i) := t$; $d(V_i) := t_f$;

// Look for a berthing position for the vessel
$insert$:= PositionBerth(V_i, W);
if $insert$ **then**
  // Once V_i is scheduled, try to assign it more QCs
  AddingCranes(V_i, V_{in});
end if
return $insert$;

After determining this first number of QCs, we must determine if there is enough continuous length and assign a berthing position. At this moment, the length of the safety distance between two contiguous vessels is taken into account (`safeLength`). Every space between each two vessels is examined and among all the possible positions, the one chosen will be the closest to the ends of the berth. This strategy is followed because if vessels are moored at these positions, the incoming vessels will have more contiguous available length each time that a vessel departs.

Then, if the vessel V_i has QCs and the length available to get moored at time t, more QCs are assigned in order to reduce the vessel service time. This process is carried out by the *AddingCranes* function (Algorithm 4) and is based on obtaining the period of time between $(m(V_i), d(V_i))$ in which there is at least one available QC without reaching the limit of QCs (`maxQC`) assigned to V_i. Once a period (t_j, t_k) has been found, the *AssignQCtoHold* function (Algorithm 5) searches among the holds assigned to QCs to determine which hold H carried out by the QC k begins later and can be completed in the given interval (*start*, *end*). The selected hold H could be moved to the list of tasks of an already assigned QC to V_i (steps 19 to 27), or a new QC could be assigned to V_i to work on this selected hold H (steps 28 to 47). In either case, if any QC becomes idle because it has

Algorithm 3 `HandlingTime` function. Distribute the holds among the available QCs

Input: V_i: Vessel to allocate; nQC: number of QCs;
Output: V_i with the holds allocated to the QCs;

// Calculate the handling time required of each hold
$T \leftarrow \{\frac{c_j(V_i)}{\texttt{movsQC}} \mid 1 \le j \le h(V_i)\}$;
Sort(T); // Sort the values of T in descending order
$q(V_i) := nQC$;

// Allocate the holds with more containers to each QC
for $j \leftarrow 1$ **to** nQC **do**
    $st_j(V_i) := m(V_i)$;
    $ht_j(V_i) := \lceil T_j \rceil$;
    $h_j(V_i) \leftarrow \{\lceil T_j \rceil\}$;
end for

// Do a greedy allocation for the rest of holds
for $j \leftarrow nQC + 1$ **to** $h(V_i)$ **do**
    // choose the QC which ends earlier
    $q_m :=$ argmin$(st_j(V_i) + ht_j(V_i),\ 1 \le j \le nQC)$;
    $h_{q_m}(V_i) \leftarrow h_{q_m}(V_i) \cup \{T_j\}$;
    $ht_{q_m}(V_i) := ht_{q_m}(V_i) + \lceil T_j \rceil + \mathtt{HH}_{\mathtt{QC}}$;
end for

Sort QCs in descending order according to their finish times
return V_i;

no assigned holds (steps 23 or 32), it performs the tasks of the last QC assigned to V_i, which then becomes available for the other vessels.

Finally, if the vessel V_i cannot be moored at time t, the whole process described above is repeated taking into consideration another time (t_k) to moor V_i (Algorithm 1). Each time t_k represents the moment in time that a QC finishes working on the hold of another vessel.

5 A metaheuristic method for BAP+QCAP

For the BAP+QCAP problem addressed in this paper, we developed different methods, which allow us to compare their results. First, different rules R have been developed following different criteria. Algorithm 6 shows the schema to schedule all the incoming vessels according to a specific rule R. Following the order given by the rule R, all vessels are chosen one by one to be moored. Each scheduled vessel is added to the set of moored vessels V_m. Generally, a vessel can be allocated at time t when there is no vessel moored in the berth or there is available contiguous quay length as well as enough free QCs to be assigned.

Algorithm 4 `AddingCranes` function. Insert another QC to V_i

Input: V_i: Vessel to insert QCs; V_m: Vessels already moored;
Output: V_i: Vessel with QCs reallocated.

// Obtain the period of time (t_j, t_k) that at least there is 1 available QC
repeat
    changes:= 0;
    // Set of vessels W which are moored at the same time as V_i
    $W \leftarrow \{v \in V_m \mid d(v) > a(V_i)\ \wedge\ m(v) < d(V_i)\}$;

    // T is the set of mooring and ending times of each QC of W
    $T \leftarrow \{m(v) \mid v \in W\ \wedge\ m(v) \ge m(V_i)\}$;
    $T \leftarrow T \cup \{st_j(v) + ht_j(v) \mid v \in W,\ 1 \le j \le q(v) \wedge (st_j(v)+ht_j(v)) \ge m(V_i) \wedge (st_j(v)+ht_j(v)) < d(V_i)\}$

    // Sort the set of time units T in ascending order
    sort(T);
    for all $t_j \in T$ **do**
        if CranesWorking$(V_i, t_j) <$ `maxQC` **then**
            for all $t_k \in T \mid t_k > t_j$ **do**
                Obtain the number of available QCs (Q_a) in the interval (t_j, t_k)
                if $Q_a > 0$ **then**
                    changes := changes + AssignQCtoHold(V_i, T_j, t_k);
                else
                    // Continue with the next time unit t_j
                    break;
                end if
            end for
        end if
    end for
until changes $= 0$; // This loop is repeated while there is any change
return V_i;

The rules R implemented for its application in Algorithm 6 are:

– *FCFS*: Vessels moor according to their arrival order, thus $\forall i, m(V_i) \le m(V_{i+1})$.
– *FCMP* (First Come Maximum Priority): Similar to FCFS where the next vessel is chosen according to their arrival order but, in this case, there is no restriction with the time the vessels can moor.
– *MWWT* (Maximum Weighted Waiting Time): Each vessel is ranked according to their weighted waiting time. The vessel with the greatest value is moored first.

Algorithm 5 `AssignQCtoHold` function. Choose one hold to be assigned to a new QC

Input: V_i: vessel to moor; $start$: starting time; end: ending time;
Output: QC added (1) or not (0)

// Search a hold whose handling task is lower or equal to $end - start$;
// $last(h_j(V_i))$ points to the last hold assigned to this QC

Sort Cranes by their finish time in descending order

// Handling time of the hold
$H := \max(last(h_j(V_i))), 1 \le j \le q(V_i) \wedge last(h_j(V_i)) < (end - start)$

// QC k which works on this hold
$k := \text{argmax}(last(h_j(V_i))), 1 \le j \le q(V_i) \wedge last(h_j(V_i)) < (end - start)$

// QC m which finishes at $start$ time
$m := j \mid 1 \le j \le q(V_i) \wedge st_j(V_i) = start$;

if $H \neq \varnothing$ **then**
  // There is a hold H that fits in the given interval
  if $(start + H) \ge d(V_i) \vee (start + H) \ge (st_k(V_i) + ht_k(V_i))$ **then**
    // Move the H to another QC would not improve the actual schedule
    return 0;
  end if

  // Delete the hold from the list of tasks of QC k
  $h_k(V_i) \leftarrow h_k(V_i) - \{H\}$;
  $ht_k(V_i) := ht_k(V_i) - H - \text{HH}_{\text{QC}}$;

  if $m \neq \varnothing \wedge k \neq m$ **then**
    // There is a QC m finishing its tasks at $start$. Add task H to QC m
    $ht_m(V_i) := ht_m(V_i) + H + \text{HH}_{\text{QC}}$;
    $h_m(V_i) \leftarrow h_m(V_i) \cup \{H\}$;

    if $|h_k(V_i)| = 0$ **then**
      // As QC k becomes idle
      Reallocate the tasks of the last QC of V_i to the QC k
      $q(V_i) := q(V_i) - 1$;
    end if
  else
    // There is no QC that finishes at $start$, so a new QC will be assigned to V_i
    if $|h_k(V_i)| > 0$ **then**
      // Check whether the tasks can be joined in the same QC
      for $j \leftarrow k + 1$ **to** $q(V_i)$ **do**
        if $st_k(V_i) + ht_k(V_i) = st_j(V_i)$ **then**
          $ht_k(V_i) := ht_k(V_i) + ht_j(V_i) + \text{HH}_{\text{QC}}$;
          $h_k(V_i) \leftarrow h_k(V_i) \cup h_j(V_i)$;

          // As QC j becomes idle
          Reallocate the tasks of the last QC of V_i to the QC j
          $q(V_i) := q(V_i) - 1$;
        end if
      end for
      // Assign the new QC to vessel V_i and increase the number of QCs
      $q(V_i) := q(V_i) + 1$;
      $k := q(V_i)$;
    end if
    $st_k(V_i) := start + \text{VV}_{\text{QC}}$; $ht_k(V_i) := H$;
    $h_k(V_i) \leftarrow \{H\}$;
  end if

  // Obtain the new departure time of V_i according to the QCs assigned
  $d(V_i) \leftarrow \max_{\forall j \in q(V_i)}(st_j(V_i) + ht_j(V_i))$;

  Sort cranes by their finish time in descending order;
end if
return 1

Algorithm 6 Vessels Allocation according to rule R

Input: V_{in}: set of ordered incoming vessels; R: rule applied;
Output: V_m: set of vessels with all the resources allocated;

$V_{last} := \varnothing$;
$V_m \leftarrow \varnothing$;
while $V_{in} \neq \varnothing$ **do**
  // Next vessel according to the rule R
  $v := \text{nextByRule}(R)$;
  $t := a(v)$;
  // For FCFS rule, the earliest mooring time possible is the mooring time of the previous vessel
  if $R = FCFS$ **then**
    $t := \max(m(V_{last}), a(v))$;
  end if

  // Schedule the chosen vessel at the earliest possible time
  $inst := \text{InsertVessel}(v, t, V_m)$;
  if ! $inst$ **then**
    $T := \{st_k(v_j) + ht_k(v_j) \mid v_j \in V_m, 1 \le k \le q(v_j) \wedge st_k(v_j) + ht_k(v_j) > t\}$;
    while $t_k \in T \wedge !\, inst$ **do**
      $inst := \text{InsertVessel}(v, t_k + \text{VV}_{\text{QC}}, V_m)$;
    end while
  end if

  // Store the last scheduled vessel
  $V_{last} := v$;
  // Update the moored and the unscheduled vessels
  $V_m \leftarrow V_m \cup \{v\}$; $V_{in} \leftarrow V_{in} - \{v\}$;
end while
return V_m;

– *EWMT* (Earliest Weighted Mooring Time): Among the vessels that can moor earlier, the operator chooses the vessel with the highest priority.

In this study, these rules are compared with our new method for the Berth Allocation and Quay Crane Assignment Problem: a metaheuristic GRASP approach. This is a randomly-biased multi-start method to obtain optimized solutions of hard combinatorial problems in a very efficient way. This method consists of two phases (Procedure 7) and these two phases are performed consecutively until a termination condition is met. This termination condition is given in one of these two forms: (1) a number of iterations; or (2) a time limit. The best solution obtained in those iterations is returned as the solution for the instance.

The first phase focuses on building a solution by means of adding one element at a time. In order to choose the next element for the solution, the elements that are not moored yet are evaluated using a greedy function that indicates how a candidate contributes to the final solution. Then, a random degree (δ) determines the number of candidates that

Procedure 7 GRASP framework

Input: Max_iterations or Time Limit;
Read input()
while No Termination condition is satisfied **do**
  // Construction phase
  $S \leftarrow \varnothing$
  Evaluate the incremental costs of the candidate elements
  while S is not a complete solution **do**
    Build the restricted candidate list (RCL)
    Select an element s from the RCL randomly
    $S \leftarrow S \cup \{s\}$
    Reevaluate the incremental costs
  end while

  // Local Search phase
  $S \leftarrow \text{LocalSearch}(S)$

  Keep track of the best solution S found in BestSolution.

  Increase the number of iterations
end while
return BestSolution

Algorithm 8 Grasp metaheuristic adapted to BAP+QCAP

Input: δ: random factor; V_{in}: incoming Vessels; K: number of neighbors; I_{max}: maximum number of iterations;
Output: V_m: set of vessels with all the resources allocated;

1: iters $\leftarrow 0$;
2: $V_m \leftarrow \varnothing$;
3: **while** No Termination condition is satisfied (I_{max}) **do**
4:   // Initialize the actual schedule and the candidates
5:   $V_s \leftarrow \{\}$;
6:   $C \leftarrow V_{in}$;
7:   **while** $C \neq \varnothing$ **do**
8:     // Evaluate the incremental costs of each candidate
9:     **for all** $v_e \in C$ **do**
10:       $f_c(v_e) := 0$;
11:       MoorVessel(v_e, V_s);
12:       $V_s' \leftarrow V_{in} \cup \{v_e\}$;
13:       **for all** $v_o \in C \mid v_o \neq v_e \wedge a(v_o) \leq a(v_e)$ **do**
14:         MoorVessel(v_o, V_s');
15:         $f_c(v_e) := f_c(v_e) + ((d(v_o) - a(v_o)) \times pr(v_o))$;
16:       **end for**
17:     **end for**

18:     // Build the Restricted Candidate List
19:     $c_{inf} := \min\{f_c(e) \mid e \in C\}$;
20:     $c_{sup} := \max\{f_c(e) \mid e \in C\}$;
21:     $RCL \leftarrow \{e \in C \mid f_c(e) \leq c_{inf} + \delta \times (c_{sup} - c_{inf})\}$;

22:     // Choose a vessel randomly
23:     $v := \text{Random}(RCL)$;
24:     MoorVessel(v, V_s);

25:     // Insert v in the partial schedule
26:     $V_s \leftarrow V_s \cup \{v\}$; $C \leftarrow C - \{v\}$;
27:   **end while**

28:   // Local search phase
29:   $V_s \leftarrow \text{LocalSearch}(V_s, K)$;

30:   Keep track of the best schedule found V_s in V_m

31:   iters $\leftarrow$ iters $+1$;
32: **end while**
33: **return** V_m;

could be eligible for this random choice election. If $\delta = 1$, all the elements are eligible, and therefore this choice is completely random. If $\delta = 0$, then it results in a completely greedy search. The second phase of the GRASP metaheuristic carries out a local search algorithm in order to improve each constructed solution in the previous phase. This local search algorithm works in an iterative manner by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood [8].

The GRASP-based procedure developed for the BAP+QCAP problem is detailed in Algorithm 8. This algorithm receives as parameters: the set of incoming vessels V_{in} waiting for mooring at the berth, the random degree (δ), the number of neighbors to explore in the local search algorithm (K) and the maximum number of iterations (I_{max}). These parameters will be discussed in Sect. 6. First, all the waiting vessels V_{in} are considered as candidates C. In step 11, each one of the candidate vessels are moored within the current state (being assigned the mooring and departure times ($m(V_i)$,$d(V_i)$), the number of QCs ($q(V_i)$), and the berthing position ($p(V_i)$)); and they are evaluated according to the greedy function f_c. Given a candidate vessel v_e, the greedy function assigned to v_e is the sum of the weighted service time of each vessel v_o that is still waiting (steps 13 to 16).

According to the greedy function f_c and the random degree indicated by δ, a Restricted Candidate List (RCL) is created (step 21). Then, one vessel v is chosen randomly among the elements from the RCL to be moored and can no longer be modified (step 23). Once v is determined, this is added to the set of vessels V_s and eliminated from the candi-

Algorithm 9 *LocalSearch* function. Local Search based on Hill Climbing for BAP+QCAP

Input: S: Schedule (set of vessels) built by the construction phase; K: number of neighbors to generate;
Output: S^*: Best schedule found from the schedule S;

1: $S^* \leftarrow S$; // S^* is the current schedule. Set of vessels with all resources allocated
2: **repeat**
3: improves := false
4: // Generate K neighbors from the current schedule
5: $S' \leftarrow \varnothing$ // Best neighbour found
6: **for** $k \leftarrow 1$ **to** K **do**
7: $S_n \leftarrow \varnothing$ // Generate a new schedule S_n
8: // Considering the mooring times of the vessels in S^* as dispatching rule, generate a new schedule S_n
9: Sort the vessels in S^* by their mooring times
10: Choose randomly two vessels i, j
11: **for** $v \in S^*$ **do**
12: // Interchange i and j within the order given by schedule S^*
13: **if** $v = i$ **then**
14: MoorVessel(j, S_n)
15: **else if** $v = j$ **then**
16: MoorVessel(i, S_n)
17: **else**
18: MoorVessel(v, S_n)
19: **end if**
20: **end for**
21: Keep track of the best neighbor schedule found in S'
22: **end for**

23: **if** S' is better than S^* **then**
24: $S^* \leftarrow S'$
25: improves := true
26: **end if**
27: **until** ! improves
28: **return** S^*;

date list C (step 26). This loop is repeated until C is empty, which means that all the vessels have been moored.

The solutions given by the construction phase of the GRASP metaheuristic always obtain valid solutions: The construction phase works as a dispatching rule by choosing each time a vessel from the RCL and inserting it into the partial schedule (set of vessels already scheduled, V_s). The Algorithms of the Sect. 4 check that all the constraints are met when a vessel is scheduled such as, among others, the safety distance between every pair of vessels or the number of QCs assigned to it. Repeating this operation for each incoming vessel obtains a feasible and valid schedule.

The second phase of the GRASP metaheuristic is shown in Algorithm 9. In order to define the neighborhood structure of the local search algorithm, a dispatching rule based on the order of the vessels according to their mooring times is applied. Thereby, a neighbor of a current schedule is created by means of interchanging (randomly chosen) two vessels in the dispatching rule (steps 9 to 20). This local search, based on the hill climbing technique, starts with the set of vessels V_s with all the resources allocated (step 1) as the current schedule S^*. K schedules from the neighborhood of the current schedule S^* are generated (step 6). If the best obtained neighbor schedule S' outperforms the current schedule S^* (step 23), according to the objective function T_w, then the current schedule S^* is replaced by S'. This loop is repeated until there is no neighbor schedule better than the current schedule (steps 2 to 27).

According to the GRASP metaheuristic framework, this search is repeated according to the number of iterations or to the time limit specified by the user. The best solution found according to the objective function T_w (V_m) is returned as the solution for the given instance of the problem.

6 Evaluation

Several experiments have been performed with two different corpus: *Dens* and *Spar*. *Spar* means that the arrival time between two vessels is sparsely distributed, and *Dens* means that the arrival time between two vessels is densely distributed. Each corpus contains 100 instances generated randomly, each one composed of a queue from 5 to 20 vessels. The terminal operators gave us two inter-arrival distributions (exponential with parameters $\lambda_{Dens} = \frac{1}{2}$ and $\lambda_{Spar} = \frac{1}{5}$, and poisson with $\lambda_{Dens} = 1.5$ and $\lambda_{Spar} = 3$ distributions) for each corpus in order to generate the arrivals for the incoming vessels. The number of required movements and length of vessels are randomly generated between 100 and 1000 containers, and between 100 and 500 meters, respectively. In all cases, the berth length (L) is fixed to 700 meters; the number of Quay Cranes is 7 (corresponding to a determined MSC berth line) and the maximum number of QCs per vessel is 5 (`maxQC`); the safety distance between QCs (`safeQC`) is 35 meters and the number of movements that QCs carry out is 25 (`movsQC`) per time unit. The time needed for the QCs to move to another hold is 5 time units ($\mathtt{HH}_{\mathtt{QC}}$), and 15 time units to another vessel ($\mathtt{VV}_{\mathtt{QC}}$). These values were estimated by the terminal operators. Without loss of generality, all the experiments were conducted assuming $\gamma = 1$.

As we mentioned above, our goal is to minimize the total weighted waiting time elapsed to serve the set of n incoming vessels. A personal computer equipped with a Intel Core 2 Q9950 2.83 GHz with 4 GB RAM was used in all the experiments.

Focusing on the static QCAP model, Fig. 9 shows the objective function (T_w) and the computation times obtained by the GRASP metaheuristic by varying the parameter K of the local search. This experiment was carried out for the *Dens* corpus with an exponential inter-arrival distribution of arrivals. Using just the constructive phase of the GRASP metaheuristic ($K = 0$), the best value achieved was 1322.7 with $\delta = 0.2$ (see Fig. 9(a)). In general, the greater the K value, the better T_w values since a deeper search in the neighborhood is carried out. For instance, the T_w obtained by $\delta = 0.2$ decreased to 1127 when $K = 14$ neighbors are generated in each step of the local search. However, we can see that for $K > 12$, we did not achieve a significance improvement in the objective function. Furthermore, it is important to note that the greater the K value for the local search, the greater the computation time. Given the $\delta = 0.2$, the computation time increased from 8.23 ms up to 15.97 ms per iteration. Therefore, a value $K = 12$ was set for the local search phase of the GRASP metaheuristic for all the following experiments.

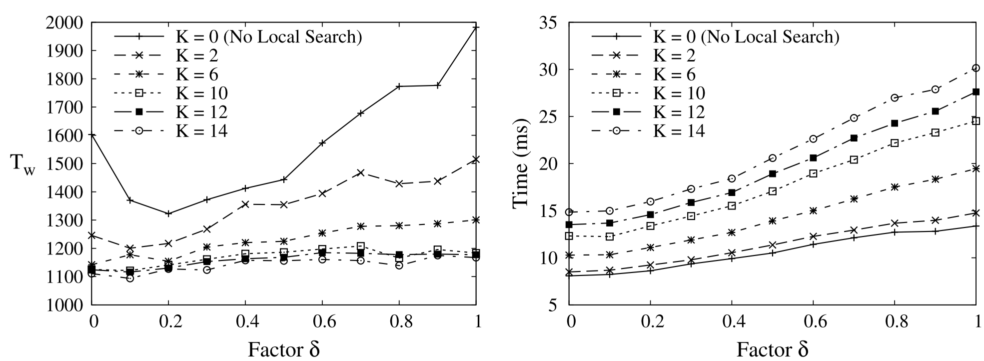
In Table 2, the dispatching rules detailed in Sect. 5 are evaluated. Each row represents the average T_w obtained by a rule on each corpus using the static QCAP model. The *EWMT* rule turned out to be the one that achieves the best results in three out of the four corpus studied. Thus, this rule will be used as a baseline for our GRASP metaheuristic algorithm.

The GRASP-based metaheuristic developed was compared with the *EWMT* rule using the two models presented: static and dynamic QCAP. Figure 10 shows the average values for the objective function (T_w) to allocate 10 vessels with an exponential inter-arrival distribution over the two corpus. As expected, the dynamic model obtained better solutions than the static one. For instance, for the *Dens* corpus (see Fig. 10(a)), for $\delta = 0.2$, the value of T_w in the static QCAP model was 315.82 and decreased to 260.96 in the dynamic QCAP model. Moreover, it can be observed that the solutions given by the GRASP method always outperformed the *EWMT* solution in the two models, specially with δ values close to 1.0 for both the *Dens* and *Spar* corpus.

Figure 11 shows the average T_w values obtained by the *EWMT* rule and the GRASP metaheuristic in the dynamic

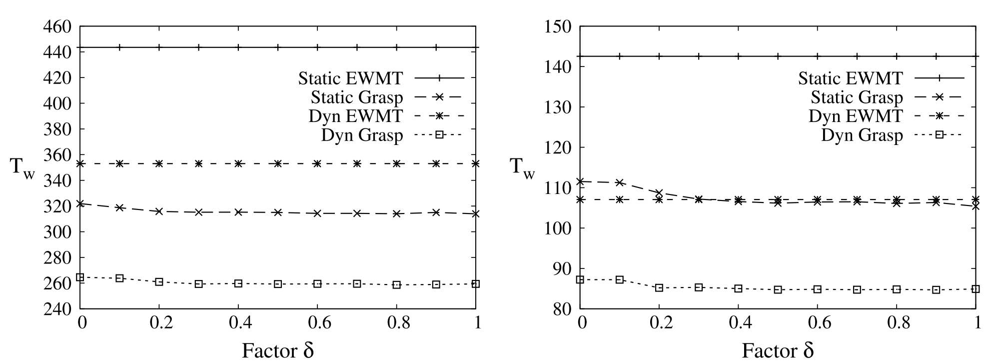
Table 2 Average T_w values for the rules and GRASP in the static QCAP model (20 vessels)

	Exp-Dens	Exp-Spar	Poisson-Dens	Poisson-Spar
FCFS	2222.26	484.45	2668.92	1116.9
FCMP	1973.21	443.26	2414.39	922.52
MWWT	2047.52	477.82	2473.57	1042.53
EWMT	1939.64	370.41	2427.8	841.79
GRASP	1414.75	273.06	1762.22	606.25



(a) Values T_w (b) Computation times

Fig. 9 Local search depending on the k value (*Dens* corpus with exponential inter-arrival distribution)



(a) Dense inter-arrival times (b) Sparse inter-arrival times

Fig. 10 T_w for 10 vessels (Exponential)

QCAP model. This experiment was carried out with instances of 20 incoming vessels with an exponential inter-arrival distribution of the *Dens* corpus using a different number of iterations for the GRASP method. It can be observed that as the number of iterations increased, the quality of our GRASP method also increased. For instance, for $\delta = 0.1$, T_w was 1135.4 with 100 iterations, while T_w decreased to 1110.21 with 400 iterations.

In Fig. 12, the same evaluation was carried out for a queue of 20 vessels with the exponential and poisson inter-arrival distributions, respectively. The same tendency as in Fig. 10 can be observed, but, in this case, $\delta \in [0.1 - 0.3]$ got the lowest values for both inter-arrival time distributions in both corpus (*Dens* and *Spar*). Moreover, the GRASP metaheuristic improved the average results given by the *EWMT* rule for each corpus. Taking into account the dynamic QCAP model, the results were improved by about 22 %–25 % with respect to the dynamic *EWMT* rule; and, in the static QCAP model, they were improved by about 26 %–27 % with respect to the static *EWMT*. These figures also show how the dynamic QCAP model always outperformed the static one. For instance, considering the poisson inter-arrival distribution and $\delta = 0.1$ (see Fig. 12(d)), the average value of T_w for the dynamic and the static QCAP model were 606.25 and 388.24, respectively.

Table 3 shows the evolution of the rules with the static QCAP model (solutions that would be provided by terminal operators) against GRASP with the dynamic QCAP model varying the number of incoming vessels from 5 vessels up

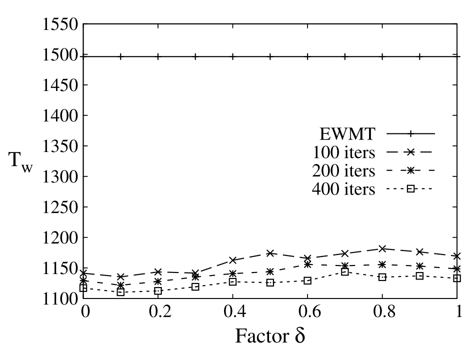


Fig. 11 T_w depending on the number of iterations in GRASP

to 20 vessels with an exponential inter-arrival distribution both for the *Dens* and *Spar* corpus. For each number of incoming vessels, the GRASP metaheuristic outperformed the average results given by the rules. For instance, given the *Spar* corpus and 15 vessels, the best rule obtained 249.41 whereas GRASP achieved 136.75. It is important to note that GRASP decreases the objective function stronger with the *Dens* corpus (see Fig. 13), since given the characteristics of the *Spar* corpus, the optimal solutions are close to the arrival order of the vessels.

Figure 14 shows the average computation times per iteration of the GRASP metaheuristic for the two models: static and dynamic QCAP. This experiment was performed for the exponential inter-arrival distribution both for the *Spar* (see Fig.14(b)) and *Dens* (see Fig. 14(a)) corpus. As the optimal solutions in the *Spar* corpus are close to the arrival order of the vessels, the average computation times for the *Spar* corpus are lower than the *Dens* corpus. Furthermore, the average time per iteration depends on the δ factor chosen since the size of the RCL is related to this parameter. Thereby, taking into account the dynamic QCAP model, this average time per iteration varied from 10.5 ms ($\delta = 0$) up to 30 ms ($\delta = 1$) for the *Spar* corpus; and, it varied from 23.2 ms ($\delta = 0$) up to 48.5 ms ($\delta = 1$) for the *Dens* corpus.

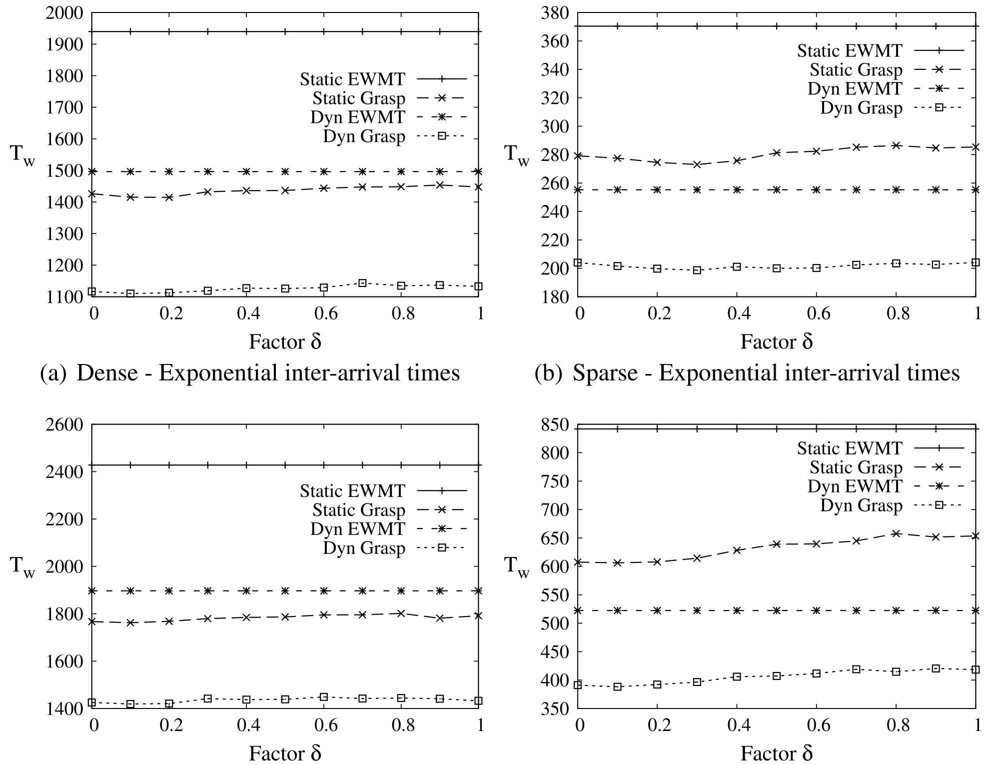
As mentioned in Sect. 3, the performance of container terminals is also evaluated according to the berth usage (B_u) (see Table 4). Table 4(a) shows the relationship between the berth usage (B_u) and the weighted waiting time (T_w) for a queue of 20 incoming vessels from the *Dens* corpus with an exponential inter-arrival distribution. In this case, only the dynamic model is considered since it obtained the best results in the previous experiments. Note that the lower T_w is, the greater the berth usage is. A value of 71.43 % was achieved for $\delta = 0.2$. Furthermore, having evaluated the dynamic and static QCAP models (see Table 4(b)), the dynamic QCAP model always achieved a better berth usage of the quay, approx. 1.34 % in average.

Another key factor studied is the QC throughput (T_{qc}). Table 5 shows that when holds are taken into account in the model (dynamic QCAP), T_{qc} is considerably improved for both *Dens* and *Spar* corpus. In other words, QCs spend less time to perform the same number of movements, e.g. with

Table 3 Average T_w values for the rules and GRASP (Exponential)

	(a) Dense inter-arrival times				(b) Sparse inter-arrival times			
	5	10	15	20	5	10	15	20
FCFS	93.3	488.86	1190.86	2222.26	51.04	176.13	312.36	484.45
FCMP	85.66	448.55	1066.46	1973.21	49.65	167.29	296.87	443.26
MWWT	92.28	477.81	1117.59	2047.52	54.62	183.29	319.48	477.82
EWMT	85.67	443.51	1039.34	1939.64	43.34	142.54	249.41	370.41
GRASP	58.92	258.65	597.45	1110.21	29.1	84.69	136.75	198.67

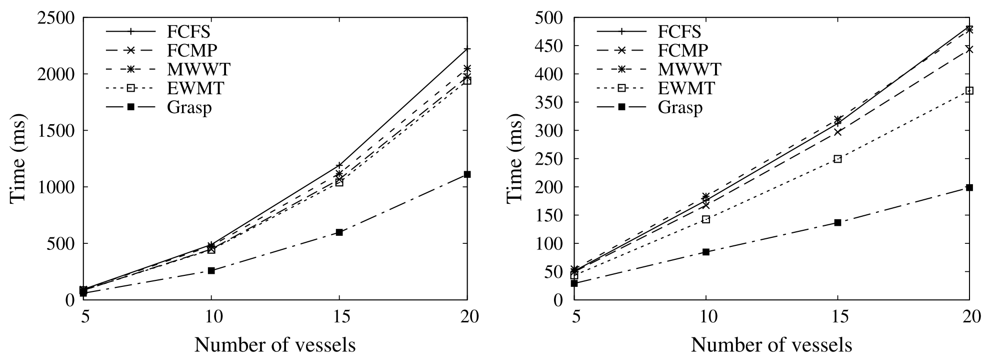
Fig. 12 Weighted waiting time for 20 vessels in sparse and dense corpus with exponential and poisson inter-arrival distribution



(a) Dense - Exponential inter-arrival times (b) Sparse - Exponential inter-arrival times

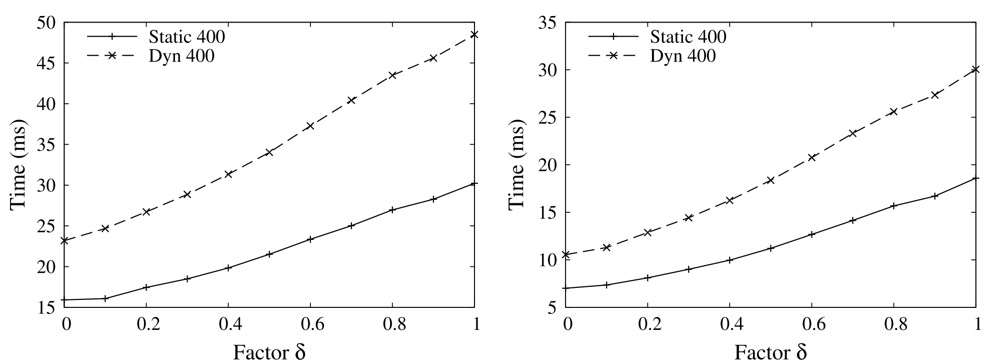
(c) Dense - Poisson inter-arrival times (d) Sparse - Poisson inter-arrival times

Fig. 13 Average T_w values for the rules and GRASP (Exponential)



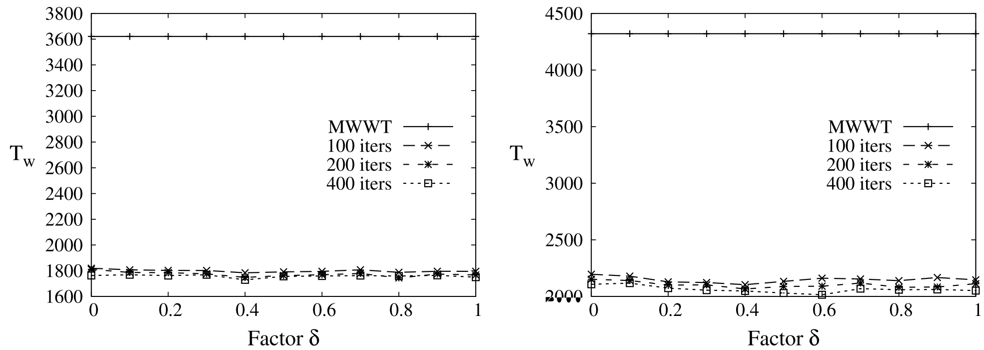
(a) Dense inter-arrival times (b) Sparse inter-arrival times

Fig. 14 Average computation times for exponential inter-arrival distribution



(a) Dense inter-arrival times (b) Sparse inter-arrival times

Fig. 15 T_w for the real data given by port operators



(a) Dynamic QCAP model (b) Static QCAP model

Table 4 B_u as a key factor in the container terminal

Factor δ	0	0.2	0.4	0.6	0.8	1
(a) Relationship between B_u and T_w (dynamic QCAP model)						
B_u %	0.7136	0.7143	0.7064	0.7065	0.7042	0.7069
T_w	1116.97	1112.41	1127.21	1129.11	1134.88	1133.04
(b) Differences in B_u between Dynamic and Static QCAP models						
Dynamic B_u %	0.7136	0.7143	0.7064	0.7065	0.7042	0.7069
Static B_u %	0.7009	0.7026	0.7047	0.6941	0.6962	0.6965

Table 5 Average time that QCs are busy

(a) Dense inter-arrival times			(b) Sparse inter-arrival times		
$\|V\|$	Static QCAP	Dynamic QCAP	$\|V\|$	Static QCAP	Dynamic QCAP
5	122.06	101.05	5	127.59	104.38
10	240.36	202.02	10	241.36	198.41
15	359.98	303.08	15	361.01	296.6
20	482.27	406.68	20	473.45	388.56

20 vessels in Table 5(a), the T_{qc} was 482.27 in the static QCAP model, whereas in the dynamic model was 406.68. Therefore, the dynamic QCAP model allows better use of the QCs, since they can be used in other vessels immediately.

Finally, we remark that the GRASP metaheuristic search has also been applied to real data given by port operators from MSC where each instance consists of 15 incoming vessels. Figure 15 shows the average T_w values. For these experiments, the rule employed was *MWWT* since it obtained the best average results, and our GRASP method was able to reduce those T_w values in both models by approximately 53 %. Comparing both models, the dynamic QCAP model reduced the T_w values by approximately 15.6 % over the static model given the same δ factor ($\delta = 0.4$).

7 Conclusions

We present a new process for allocating berth space for a number of vessels that uses the well-known GRASP metaheuristic. The developed method also adds the Quay Crane Assignment Problem into the model, taking into account the holds of each incoming vessel. The holds of the vessels are introduced in the dynamic QCAP model. The proposed GRASP metaheuristic has been compared to usual scheduling methods employed in container terminals (FCFS, FCMP, MWWT, EWMT). It can be observed how this metaheuristic reduces the waiting time and increases both the berth utilization and the throughput of QCs. These benefits are even greater when the dynamic QCAP model is employed since QCs are assigned in a more efficient way.

Due to the continuous increase of vessels traffic, our proposed metaheuristic could be employed since the difference between the GRASP-based method and the usual dispatching rules is becoming more and more significant. Therefore, allocation methods currently used in container terminals can be improved to a great extent by integrating metaheuristic approaches from areas of artificial intelligence.

BAP+QCAP solutions are executed in dynamic real-world environments where incidences can occur. Thus, an initial schedule might become invalid due to some incidences such as breakdowns in the QC engines, delays in the arrival of the vessels or deviations from the input data given by the shipping companies. Two main approaches are usually applied to manage these incidences: proactive and reactive [18]. The aim of a proactive approach is to obtain robust schedules that remain valid against incidences. A reactive approach gives rise to the process of re-scheduling. These issues are interesting and open questions in real applications.

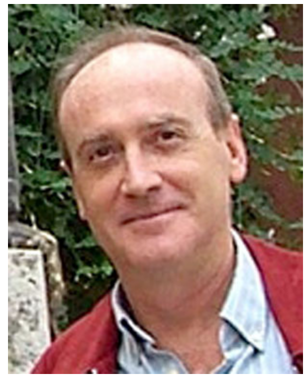
Acknowledgements This work has been partially supported by the research projects TIN2010-20976-C02-01 (Ministerio de Ciencia e Innovación, Spain) the fellowship program FPU (AP2010-4405), and also with the collaboration of the maritime container terminal MSC (Mediterranean Shipping Company S.A.).

References

1. Ayvaz D, Topcuoglu H, Gurgen F (2012) Performance evaluation of evolutionary heuristics in dynamic environments. Appl Intell 37(1):130–144
2. Bierwirth C, Meisel F (2010) A survey of berth allocation and quay crane scheduling problems in container terminals. Eur J Oper Res 202(3):615–627
3. Cheong C, Tan K, Liu D (2009) Solving the berth allocation problem with service priority via multi-objective optimization. In: IEEE symposium on computational intelligence in scheduling, 2009, CI-sched '09, pp 95–102
4. Christiansen M, Fagerholt K, Ronen D (2004) Ship routing and scheduling: status and perspectives. Transp Sci 38(1):1–18
5. Consultants DS (2010) Global container terminal operators annual review and forecast. Annual Report
6. Cordeau J, Laporte G, Legato P, Moccia L (2005) Models and tabu search heuristics for the berth-allocation problem. Transp Sci 39(4):526–538
7. Daganzo C (1989) The crane scheduling problem. Transp Res, Part B, Methodol 23(3):159–175
8. Feo T, Resende M (1995) Greedy randomized adaptive search procedures. J Glob Optim 6(2):109–133
9. Festa P, Resende MG (2009) An annotated bibliography of grasp–part ii: applications. Int Trans Oper Res 16(2):131–172
10. Giallombardo G, Moccia L, Salani M, Vacca I (2010) Modeling and solving the tactical berth allocation problem. Transp Res, Part B, Methodol 44(2):232–245
11. Guan Y, Cheung R (2004) The berth allocation problem: models and solution methods. OR Spektrum 26(1):75–92
12. Henesey L (2006) Overview of transshipment operations and simulation. In: MedTrade conference, Malta, April 2006, pp 6–7
13. Imai A, Nagaiwa K, Tat C (1997) Efficient planning of berth allocation for container terminals in Asia. J Adv Transp 31(1):75–94
14. Imai A, Chen H, Nishimura E, Papadimitriou S (2008) The simultaneous berth and quay crane allocation problem. Transp Res, Part E, Logist Transp Rev 44(5):900–920
15. Kim K, Günther H (2006) Container terminals and cargo systems. Springer, Berlin
16. Kim KH, Park YM (2004) A crane scheduling method for port container terminals. Eur J Oper Res 156(3):752–768
17. Lai KK, Shih K (1992) A study of container berth allocation. J Adv Transp 26(1):45–60
18. Lambrechts O, Demeulemeester E, Herroelen W (2008) Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. J Sched 11(2):121–136
19. Lee D, Wang H, Miao L (2008) Quay crane scheduling with non-interference constraints in port container terminals. Transp Res, Part E, Logist Transp Rev 44(1):124–135
20. Lee DH, Chen JH, Cao JX (2010) The continuous berth allocation problem: a greedy randomized adaptive search solution. Transp Res, Part E, Logist Transp Rev 46(6):1017–1029
21. Liang C, Huang Y, Yang Y (2009) A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning. Comput Ind Eng 56(3):1021–1028
22. Lim A (1998) The berth planning problem. Oper Res Lett 22(2–3):105–110
23. Liu J, Wan YW, Wang L (2006) Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. Nav Res Logist 53(1):60–74
24. Meisel F, Bierwirth C (2009) Heuristics for the integration of crane productivity in the berth allocation problem. Transp Res, Part E, Logist Transp Rev 45(1):196–209
25. Mohi-Eldin E, Mohamed E (2010) The impact of the financial crisis on container terminals (a global perspectives on market behavior). In: Proceedings of 26th international conference for seaports & maritime transport
26. Park Y, Kim K (2003) A scheduling method for berth and quay cranes. OR Spektrum 25(1):1–23
27. Peterkofsky R, Daganzo C (1990) A branch and bound solution method for the crane scheduling problem. Transp Res, Part B, Methodol 24(3):159–172
28. Rodríguez-Molins M, Salido MA, Barber F (2010) Domain-dependent planning heuristics for locating containers in maritime terminals. In: Proceedings of the 23rd international conference on industrial engineering and other applications of applied intelligent systems. LNCS, vol 6096. Springer, Berlin, pp 742–751
29. Rodriguez-Molins M, Salido M, Barber F (2012) Intelligent planning for allocating containers in maritime terminals. Expert Syst Appl 39(1):978–989
30. Salido M, Sapena O, Barber F (2009) An artificial intelligence planning tool for the container stacking problem. In: Proceedings of the 14th IEEE international conference on emerging technologies and factory automation, pp 532–535
31. Salido MA, Rodriguez-Molins M, Barber F (2012) A decision support system for managing combinatorial problems in container terminals. Knowl-Based Syst 29:63–74
32. Stahlbock R, Voß S (2008) Operations research at container terminals: a literature update. OR Spektrum 30(1):1–52
33. Steenken D, Voß S, Stahlbock R (2004) Container terminal operation and operations research-a classification and literature review. OR Spektrum 26(1):3–49
34. Szlapczynski R, Szlapczynska J (2012) On evolutionary computing in multi-ship trajectory planning. Appl Intell 37:155–174
35. Theofanis S, Boile M, Golias M (2009) Container terminal berth planning. Transp Res Rec 2100:22–28
36. ValenciaPort F (2009) Automation and simulation methodologies for assessing and improving the capacity, performance and service level of port container terminals. Ministerio de Fomento (P19/08), Spain
37. Vis I, De Koster R (2003) Transshipment of containers at a container terminal: an overview. Eur J Oper Res 147:1–16

Mario Rodriguez-Molins is a Ph.D. student at Universitat Politècnica de València (UPV), Spain and holds a fellowship program FPU supported by the Spanish government until 2015. He received the B.S. and M.S. degrees in computer science from UPV in 2009 and 2011, respectively. His current work focuses on scheduling, planning, metaheuristics and multi-objective optimization.

Miguel A. Salido is Associated Professor in the Department of Computer Science at Universitat Politècnica de València. His expertise area is focused on constraint programming and its application to planning and scheduling problems. He is the author of more than 80 papers published in international journals and conferences. He is PC member of international conferences in the area: IJCAI, ECAI, ICAPS. He has participated in several national and European research projects. More information can be found at: http://users.dsic.upv.es/~msalido/.



Federico Barber is Full Professor in the Department of Computer Science, where he leads a research team in artificial intelligence. He has worked on the development of temporal reasoning systems, Constraint Satisfaction Problems, planning and scheduling. He is the author of more than 100 papers in international journals and conferences. His research has produced several tools for solving real-world optimization combinatory problems. He has participated in and led several national and European research projects related to these areas, and has led several technology transfers to relevant companies. He is the current president of the IberoAmerican Society of Artificial Intelligence (IBERAMIA) and member of several scientific associations.