A Biased Random-Key Genetic Algorithm for the time-invariant Berth Allocation and Quay Crane Assignment Problem

Juan Francisco Correcher, Ramon Alvarez-Valdes,

University of Valencia, Department of Statistics and Operations Research, Doctor Moliner 50, 46100 Burjassot, Valencia, Spain

e-mail addresses:
juan.correcher@uv.es (J.F. Correcher);
ramon.alvarez@uv.es (R. Alvarez-Valdes).

Abstract

Maritime transportation plays a crucial role in the international economy. Port container terminals around the world compete to attract more traffic and are forced to offer better quality of service. This entails reducing operating costs and vessel service times. In doing so, one of the most important problems they face is the Berth Allocation and quay Crane Assignment Problem (BACAP). This problem consists of assigning a number of cranes and a berthing time and position to each calling vessel, aiming to minimize the total cost. An extension of this problem, known as the BACAP Specific (BACASP), also involves determining which specific cranes are to serve each vessel. In this paper, we address the variant of both BACAP and BACASP consisting of a continuous quay, with dynamic arrivals and time-invariant crane-to-vessel assignments. We propose a metaheuristic approach based on a Biased Random-key Genetic Algorithm with memetic characteristics and several Local Search procedures. The performance of this method, in terms of both time and quality of the solutions obtained, was tested in several computational experiments. The results show that our approach is able to find optimal solutions for some instances of up to 40 vessels and good solutions for instances of up to 100 vessels.

Keywords: Container terminal, Berth Allocation, Quay Crane assignment, Metaheuristic, Genetic Algorithm, Local Search.

1 Introduction

Since the 1960s, the traffic of goods by means of standardized containers has become widespread and specialized facilities and cargo ships have been developed to manage them efficiently. According to UNCTAD (2016), more than 175 million TEUs (Twenty-foot Equivalent Units) were moved in 2015, while in the decade 2006–2015 the number of container movements increased by more than 55%. This evidences the high impact that containerized maritime commerce continues to attain worldwide.

Container terminals are the special docks and facilities dedicated to the management of container traffic at seaports. Due to the pressures of competition in international trade, they attempt to improve their processes to reduce costs, enhance their capacity, and offer better quality of service to their clients. Therefore, they are interested in optimizing the allocation of the berth positions to calling vessels, the use of the quay cranes, the container stacking procedures, and other operational processes. The efficient management of such operations is the key to reducing costs, avoiding contractual penalties, and decreasing waiting and processing times of vessels.

The main areas in a container terminal are the seaside, the yard, and the landside. The first is the quay, where the vessels are moored and the quay cranes load and unload the containers. The yard is the place where containers are stacked up temporarily, while the landside is the interchange area where trucks and trains carry containers to and from the hinterland. Several overviews of container terminal operations have been presented by Stahlbock and Voß (2008), Rashidi and Tsang (2013), and Li et al. (2015).

One of the most important problems at the seaside is the *Berth Allocation Problem* (BAP), the problem of determining a minimum-cost plan for the assignment of berthing times and positions to calling vessels. The cost includes the waiting time before berthing, the delay in the departure date, and the deviation from the desired position at the quay. Several versions of this problem have been proposed depending on the type of berth considered (discrete, continuous, or hybrid), the arrival time of vessels (dynamic, static), their processing times (fixed or depending on the number of cranes), the different costs involved, and other factors. The most up-to-date reviews of the problems and approaches related to seaside operations were presented by Bierwirth and Meisel (2010, 2015).

The standard BAP considers the processing times to be given in advance as an estimate, thus assuming a predetermined number of cranes assigned to each vessel. However, the number of cranes is a variable that influences the berth allocation, and therefore the optimization process. For this reason, many researchers now address the integrated problem called *Berth Allocation and quay Crane Assignment Problem* (BACAP). In this problem, as well as assigning a berthing time and position, a number of cranes also has to be assigned to each vessel, thereby determining its processing time.

The crane assignment defines two variants of the problem. The number of cranes assigned to each vessel can be time-invariant, i.e., fixed during the entire processing of the vessel, or variable-in-time, allowing the number of cranes to vary in each time period during its processing. The time-invariant version is easier to handle for human operators and does not require minimizing the number of crane movements. By contrast, the variable-in-time version can achieve better crane utilization at the expense of putting great pressure on dock operations and making the plan more complicated and sensitive to unexpected incidents.

A further approximation to reality is attained by determining not only the number of cranes, but also which specific cranes have to serve each vessel. This problem is known as the BACAP Specific or BACASP and is attracting increasing attention in the field. As we will see in the literature review, one of the limitations in time-invariant BACAP and BACASP research is the size of the instances that proposed methods are able to address when considering a continuous quay and dynamic vessel arrival. This is mostly due to the lack of metaheuristic approaches tailored to this version of the problem and the common limitations of mathematical programming approaches, which up to now have not been able to solve instances with more than 50 vessels within a planning horizon of one week in reasonable time (Correcher et al., 2017). Hence, the main motivation of this study is to propose a metaheuristic approach capable of obtaining good solutions regardless of the instance size.

In this paper we address both the time-invariant BACAP and BACASP with continuous quay, dynamic arrival and, for each vessel, an estimate of its processing time for each potential number of cranes admitted. We propose a new Biased Random-Key Genetic Algorithm (BRKGA) with Memetic improvement and Local Search (LS) to obtain good solutions in short computational time for instances involving up to 100 vessels in a planning horizon of one week.

The remainder of this paper is organized as follows. In Section 2, the previous studies on the problem are reviewed, and in Section 3 we describe the BACAP variant addressed in this paper. In Section 4 we present the Genetic Algorithm, the constructive algorithm, and the Local Search procedures. The BACASP is explained as an extension of the BACAP in Section 5, where we also specify the changes needed to adapt the algorithms to address it. In Section 6 we describe the computational experiments conducted and discuss the results. Finally, in Section 7, we draw some conclusions and propose future work.

2 Literature review

In this section we review closely related studies on the continuous BACAP and BACASP. Comprehensive reviews that address the different versions of these problems, including discrete and hybrid quay variants, are the aforementioned papers of Bierwirth and Meisel (2010, 2015).

2.1 Variable-in-time crane assignment

The seminal work on the continuous BACAP was presented by Park and Kim (2003). They formulated a Mixed Integer Linear Programming model (MILP) for the problem with variable-in-time crane assignment. Moreover, they proposed a Lagrangean relaxation of the model and used the subgradient method to obtain near-optimal solutions on randomly generated instances of up to 40 vessels within a planning horizon of one week. They also presented a dynamic programming algorithm to obtain solutions for the corresponding BACASP. On the basis of this approach, Zhang et al. (2010) proposed a mixed integer linear model for a BACASP that takes into account the limited range of movement of each crane.

Meisel and Bierwirth (2009) studied a continuous BACAP with variable-in-time crane assignment and the possibility of speeding up the arrival of each vessel, incurring a cost proportional to the time advanced. They also considered decreasing marginal crane productivity as the number of cranes serving a vessel increases, due to interferences between them. The authors proposed an MILP and two metaheuristic approaches: a Tabu Search and a Squeaky Wheel Optimization. They reported good results on previous and newly generated instances of up to 40 vessels within a planning horizon of one week. Elwany et al. (2013) studied the same problem considering that water depth varies depending on the position. They extended the model of Meisel and Bierwirth (2009) and proposed a Simulated Annealing based on a heuristic capable of constructing feasible solutions from ordered lists of vessels.

More recently, Iris et al. (2015) addressed the same problem in both the variable-in-time and time-invariant versions. They proposed several novel set partitioning formulations and some variable reduction techniques. They compared with Meisel and Bierwirth (2009) on the same instances and reported several improvements on previous results. Raa et al. (2011) and Xiao and Hu (2014) proposed several MILPs for rolling time horizon schemes, and Hu (2015) also considered a rolling horizon with the novelty of periodic balancing utilization of the quay cranes.

Türkoğullari et al. (2016) proposed both an MILP for the deeply integrated variable-in-time BACASP and a cutting plane algorithm based on a decomposition scheme. The results reported show that their method could solve to optimality instances of up to 60 vessels within a planning horizon of 600 hours. Han et al. (2015) addressed a variable-in-time BACAP taking into consideration the ranges of movement of the quay cranes. They proposed an MILP for the BAP, another for the quay crane assignment problem (QCAP), and a Particle Swarm Optimization heuristic. A related approach was proposed by Karam and Eltawil (2016), who developed a functional integration of two independent models for BAP and QCAP able to obtain good solutions on instances of up to 21 vessels arriving within an horizon of 168 hours.

A further approximation to reality was achieved by Rodriguez-Molins et al. (2014a), who considered the moving time of cranes along the serving vessel and along the quay in both the variable-in-time and the time-invariant BACASP. They proposed a GRASP that was applied on real-life instances of up to 20 vessels. Rodriguez-Molins et al. (2014b) also proposed a multi-objective MILP and a Multi-Objective Genetic Algorithm with Simulated Annealing to perform robust scheduling on realistic instances. Likewise, Shang et al. (2016) considered the setup times of quay cranes, including them in the mathematical formulation. They proposed a Genetic Algorithm and three MILPs: a basic model, a robust model capable of facing uncertainties, and a version of the latter with price constraints. These methods were tested on the instances of Meisel and Bierwirth (2009), in which they obtained optimal solutions on those involving up to 20 vessels.

Chang et al. (2010) introduced the energy consumption of quay cranes in a multi-objective MILP and a Parallel Genetic Algorithm tailored to address a rolling horizon scheme for the BACASP. They reported good results on instances of up to 40 vessels within a planning horizon of 72 hours. Hu et al. (2014) also included fuel consumption and emissions incurred by vessels and quay cranes in a new multi-objective programming model. Good solutions were obtained on instances with up to 20 vessels within the same planning horizon. More recently, He (2016) also considered energy consumption in the BACAP and proposed an MILP and a Memetic Algorithm capable of obtaining both optimal solutions on instances with up to 24 vessels and good solutions on those with up to 40 vessels.

Beyond that, Meisel and Bierwirth (2013) integrated the main optimization problems that appear in the seaside of container terminals (BAP, QCAP and QCSP: Quay Crane Scheduling Problem) in an iterative framework with three phases in which several MILPs are solved. It was tested on instances of up to 40 vessels within a planning horizon of one week.

2.2 Time-invariant crane assignment

Blazewicz et al. (2011) addressed for the first time a continuous BACASP with time-invariant crane assignments by formulating a moldable task scheduling problem. This approach was tested on instances containing up to 45 vessels within a time horizon of 100 hours. Yang et al. (2012) proposed a Nested Loop-based Evolutionary Algorithm for the BACASP and obtained good results compared with Park and Kim (2003). Le et al. (2012) addressed the BACASP by proposing a multi-objective MILP model and a Multi-Objective Particle Swarm Optimization.

A different approach was proposed by Chen et al. (2012), who developed a combinatorial Bender's cuts algorithm based on a previous BACASP model. They were able to obtain good solutions on instances with up to 26 vessels within a planning horizon of one week. As we have already mentioned, Iris et al. (2015) also developed a generalized set partitioning formulation for the time-invariant BACAP, while Rodriguez-Molins et al. (2014a) developed a GRASP for the time-invariant BACASP.

Türkoğullari et al. (2014) tackled both the BACAP and the BACASP with time-invariant crane assignment, for which they proposed two MILPs, an algorithm to get a BACASP solution from a BACAP solution, and a cutting plane algorithm. The authors reported that their model for the BACAP and the cutting plane algorithm for the BACASP obtained optimal solutions on instances of up to 60 vessels within a planning horizon of 600 hours. Along this line, we have also addressed these problems in a recent work by developing new MILPs that are capable of attaining optimal solutions on instances of up to 50 vessels within a planning horizon of one week (Correcher et al., 2017).

After reviewing the most recent works on the continuous BACAP and BACASP, we can conclude that the time-invariant version has received less attention than the variable-in-time one and that the largest instances addressed in the literature consist of around 40-50 vessels within a one-week planning horizon. Moreover, only a few of the methods proposed are able to produce good solutions for instances of this size. This contrasts with the actual traffic in the largest container terminals, which nowadays are required to deal with more than 50 vessels in a week. Therefore, we consider it interesting to address this issue by proposing a metaheuristic approach not restricted by the size of the problem.

In the following section, we describe the time-invariant BACAP addressed in this study, which is the same problem studied by Türkoğullari et al. (2014) and Correcher et al. (2017).

3 Description of the BACAP

The Berth Allocation and quay Crane Assignment Problem (BACAP) is the optimization problem of assigning berth position, number of cranes, and mooring time to calling vessels, minimizing the total assignment cost. We deal with the version of the problem in which there is only one continuous quay and the number of quay cranes serving a vessel, once assigned, is kept fixed during its processing. The arrival time of each vessel is known in advance, as well as the maximum and minimum number of cranes that can be assigned to the vessel and an estimate of its processing time for each number of cranes.

A berth plan can be rendered as a space-time layout (see Figure 1) where the vertical axis represents the mooring positions on the quay and the horizontal axis represents time. Each vessel is rendered as a rectangle whose base represents its processing time and the height its length. Vessels can be moored along the quay, while quay cranes can move along the quay to serve the vessels provided that they do not cross each other. For each vessel, we consider three different costs: the cost of waiting before berthing, the cost of delay after the desired departure time, and the cost of deviating from the desired position on the quay.

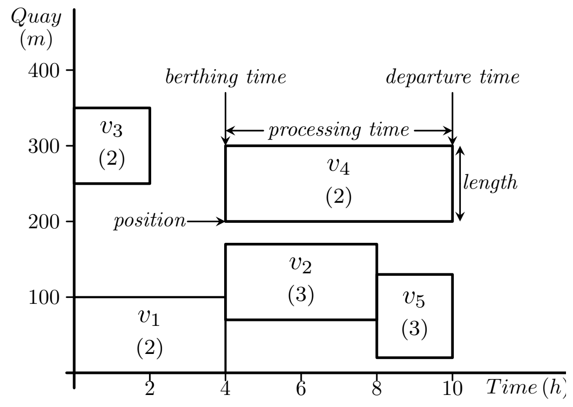


Figure 1: A solution with 5 vessels and a quay 400 m long with 5 cranes. The numbers in parentheses are the quantity of cranes assigned to each vessel

3.1 Parameters

The following parameters are assumed to be given:

- Set of time periods: $T = \{1, 2, \ldots, H, \ldots\}$, H being the planning horizon
- Length of the quay: L
- Number of quay cranes: Q
- Set of calling vessels: V; where $N = |V|$ is the number of vessels

  For each vessel i, we know:

  - length: l_i
  - expected arrival time: a_i
  - desired departure time: s_i
  - minimum and maximum number of quay cranes that can be assigned to the vessel: q_i^{min}, q_i^{max}
  - estimated processing time considering q cranes: u_i^q
  - desired position on the quay: d_i
  - cost per waiting time period for berthing: C_i^w
  - cost per delay time period: C_i^d
  - cost per unit length away from the desired position on the quay: C_i^p
  - cost per period of berthing time exceeding the time horizon: C_i^h

The desired position on the quay is usually the position closest to the location in the yard where the containers to be loaded onto/unloaded from the vessel are placed. Therefore, the deviation from the desired position is the distance between that position and the assigned berthing position. The waiting time is the difference between its berthing time and its expected arrival time, while the delay is the difference between the actual and desired departure times, if it is greater than zero.

The planning time horizon H is given in advance, but for instances with many vessels or with vessels arriving at the end of the planning interval, there may not be feasible solutions in which all the vessels are moored within it. For this reason, we also admit solutions in which one or more vessels are berthed beyond the time horizon at the expense of incurring a special cost. This cost for each vessel is proportional to the difference between its berthing time and the time horizon, if it is greater than zero. Thus, solutions with all the vessels moored within the time horizon are favoured.

3.2 Assumptions and constraints

The assumptions and constraints of the problem are as follows:

- Time
  - The planning horizon is divided into multiple equal time segments.
  - Vessels are to be moored within the planning horizon.
- Quay
  - Each position on the quay can accommodate one vessel at a time.
- Vessels
  - When a vessel is moored, the berthing position is kept fixed.
  - Once started, the handling of a vessel cannot be interrupted.
  - The handling time of each vessel is considered to be independent of its berthing position. This assumption is reasonable if the quay has enough machinery and workers for container transportation between the yard and the quay at any moment. Hence, the cranes serving each vessel do not need to wait for vehicles. The increased transportation cost produced if the position of the vessel deviates from its desired position is included in the objective function.
  - The handling time of each vessel depends on the number of cranes assigned to it. No specific relation between them is assumed, so it can be either linear or non-linear.
  - The time for docking and undocking maneuvers is considered to be included in the vessel handling time.
  - Vessels may have different relative importance. Therefore, cost coefficients are specific to each vessel.
  - The inter-ship clearance is included in the vessel length. In general, for vessels longer than 130 m, this clearance corresponds to 10% of its length. For small vessels, the minimum clearance is 10 m.
- Cranes
  - The number of cranes available at the quay is fixed and all the cranes have the same characteristics.
  - All quay cranes can move along the whole length of the quay, but they cannot cross each other.
  - The time spent by cranes when moving along the quay is considered negligible compared to the handling time of vessels.
  - Each quay crane can be assigned to one vessel at most in each time period.

– The number of quay cranes assigned to a vessel does not change during its stay at the quay.

– There is a minimum and a maximum number of cranes that can be assigned to a vessel.

3.3 Variables and objective function

For each vessel i, the decision variables are: the berthing time, t_i, the berthing position at the quay, p_i, and the number of cranes assigned, r_i. Since we intend to minimize the overall cost, the objective function is the minimization of the waiting cost, delay cost, deviation cost, and exceeding horizon cost incurred by the terminal for each vessel i:

$$Min \sum_{i \in V} (C_i^w (t_i - a_i) + C_i^d (t_i + u_i^{r_i} - s_i - 1)^+ + C_i^p |p_i - d_i| + C_i^h (t_i - H)^+) \quad (1)$$

Note that $(expression)^+$ is to be interpreted as $max(expression, 0)$.

These parameters and constraints, and this objective function define the BACAP that is solved in the next section by means of a Biased Random-keys Genetic Algorithm.

4 A Biased Random-Key Genetic Algorithm for BACAP

Previous studies have shown that even simple versions of the BAP are NP-hard (Lim, 1998), and therefore the BACAP is also NP-hard. As our objective is to obtain good solutions in short computing times for all types of instances and to have a flexible framework at our disposal, the best option is to make use of metaheuristic strategies. Evolutionary Computation algorithms, especially Genetic Algorithms, have proved to be a good approach to tackle similar Berth Allocations problems (Bierwirth and Meisel, 2015, Nishimura et al., 2001, Lalla-Ruiz et al., 2014) and closely related packing problems (Gonçalves and Resende, 2013, Hopper and Turton, 2001, Iori et al., 2003). Along these lines, we propose a Biased Random-Key Genetic Algorithm with some memetic characteristics, a constructive algorithm based on ordered lists of vessels, and several Local Search procedures.

The Genetic Algorithm can be extended to consider multiple populations, thus allowing parallel evolutions with sporadic migrations of individuals between them. The scheme of the BRKGA applied in this work is inspired by Gonçalves and Resende (2012).

4.1 The Genetic Algorithm

We consider a number N_{pop} of populations with $N_{indiv} = MultPop * N$ individuals each. That is, the number of individuals in a population depends on the number of vessels N and on a constant $MultPop$ that must be specified. Each individual has one chromosome consisting of two lists: a list of keys-to-vessels and a list of numbers of cranes-to-vessels. Each position in these lists corresponds, respectively, to the random key and the number of cranes assigned to the vessel with that index. The key is a number between 0 and 1, while the number of cranes must be between the minimum and the maximum number of cranes allowed for that vessel (Figure 2). The effective list of vessels is determined by decoding the list of random keys-to-vessels, which is done by sorting the keys in non-decreasing order and then taking their indexes (Figure 3).

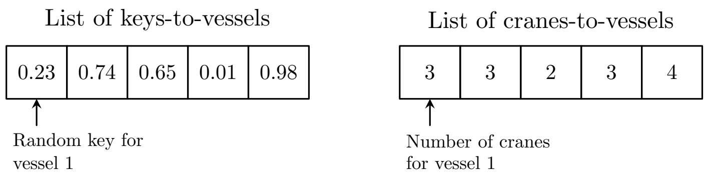


Figure 2: Chromosome of an individual for a BACAP instance with 5 vessels.

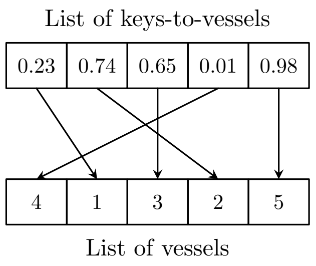


Figure 3: Decoding a list of keys-to-vessels to a list of vessels.

The cranes-to-vessels list and the sequence of vessels obtained by sorting the keys represent a solution as long as they are considered together with the constructive algorithm (Section 4.2). The constructive algorithm uses this information to generate a feasible solution by adding the vessels one by one to the berth plan. After the construction, a refinement procedure is applied, aiming at improving the solution by reducing the idle time of the cranes to a minimum (Section 4.3). It is thereby possible to obtain a new list of cranes which, used together with the list of vessels, can produce a better solution. If this is the case, the chromosome is updated with the new list and can influence the offspring. This inheritance of acquired characteristics turns the Genetic Algorithm into a kind of *Memetic Algorithm*.

The fitness of an individual is the negative of its objective function value, since it is a minimization problem, and the genetic operators are: *elite crossover*, *immigration* and *migration*. The crossover works between two parents, considering a bias $CrossBias \in [0, 1]$ in favour of the chromosome coming from the first parent, which is taken from the elite (Figure 4). The same process is applied independently over the list of cranes-to-vessels. Besides the crossover, at each generation we add new randomly generated individuals as a kind of immigration. Moreover, a migration is performed every $GensMig$ generations, adding the best individual among all the populations into the other populations in which it is not present and removing the worst individual to keep the same population size.

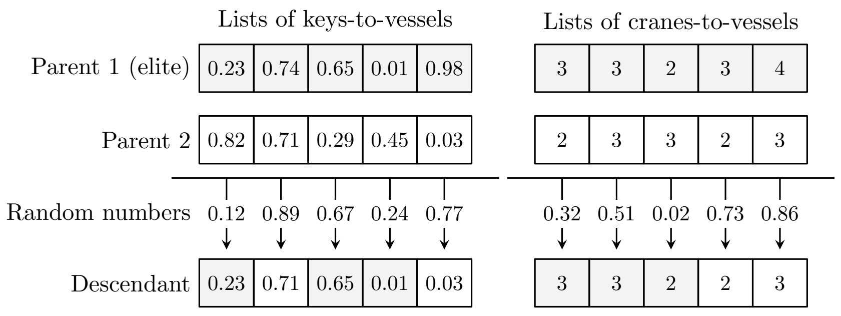


Figure 4: Example of a crossover. A gene from the first parent passes to the descendant if the random number is less than $CrossBias = 0.7$.

The initial population consists of individuals whose key-to-vessel lists are generated according to several priority rules, which are defined using elements of the problem: arrival time (a_i), length (l_i), desired departure time (s_i), maximum and minimum processing times ($u_i^{q_i^{max}}$, $u_i^{q_i^{min}}$), maximum and minimum area ($l_i u_i^{q_i^{max}}$, $l_i u_i^{q_i^{min}}$), and maximum and minimum slack ($s_i - (a_i + u_i^{q_i^{max}})$, $s_i - (a_i + u_i^{q_i^{min}})$). These rules sort the vessels in both non-increasing and non-decreasing order, one of them to produce good solutions and the other to produce diverse solutions. This provides the evolutionary process with information that may help the search, as reported in previous works on Genetic Algorithms applied to similar problems (Frojan et al., 2015). The crane lists are generated by assigning to each vessel a random number of cranes from among its allowed values. This ensures that the descendants' crane lists will be valid too. As the population size can be greater than the number of priority rules, the remaining individuals in each population are generated randomly, thereby introducing more diversity.

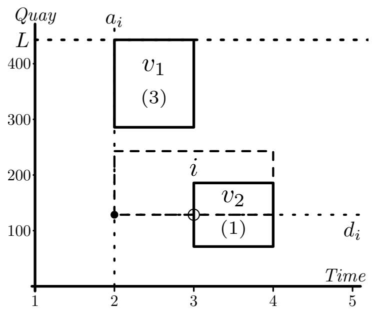
The process applied to each population to generate a new generation is as follows. First, the constructive algorithm is applied to each individual in the population to obtain its corresponding feasible solution. Then, individuals are sorted according to their fitness and the best $E\%$ individuals are marked as the elite. The following generation will be composed of this elite, $I\%$ new individuals randomly generated as immigration, and $100\% - E\% - I\%$ individuals resulting from the crossover between pairs of individuals randomly chosen, the first one being from the elite and the other from the entire population. The parent from the elite is the one favoured by the bias. Migration is performed only every $GensMig$ generations. This process is repeated until a stopping criterion is met, a time limit of $TimeLimitGen$ seconds. Afterwards, a Local Search procedure is applied to every individual in each population until a stopping criterion is met, with the aim of refining the best solutions obtained by the Genetic Algorithm (Section 4.4).

4.2 Constructive algorithm

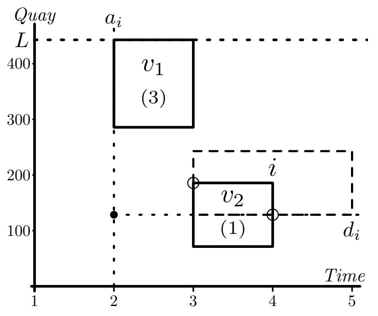
An individual represents a solution only if it is considered together with a constructive algorithm. This algorithm builds a feasible berth plan from the list of cranes-to-vessels and the list of vessels resulting from sorting the keys. We propose a fast constructive algorithm which works by assigning position, time, and number of cranes to vessels according to the order of the vessels in the list, taking them one by one. It is based on the *Exploratory Constructive Algorithm (ECA)* that we presented in a previous study (Frojan et al., 2015), with the novelty that now it also assigns a number of cranes to each vessel and checks each time that the number of cranes assigned to vessels does not exceed Q.

First we fix the number of cranes corresponding to each vessel according to the cranes-to-vessels list. Therefore, the processing time of each vessel is known and we can assign a berthing time and position to each vessel, taking them sequentially from the ordered list of vessels. The objective is to achieve the best allocation for each vessel in a berth plan in which the previous vessels in the list have already been assigned. To do this, each time we extract a vessel from the list we define a set K of candidate assignments ordered by non-decreasing cost. The set K is filled and explored throughout the process, always extracting the least-cost candidate. The first candidate assignment included in K is the one consisting of the desired position and the arrival time of the vessel.

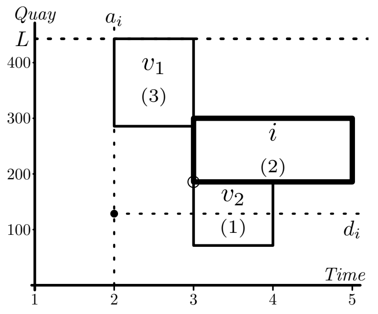
Once the least-cost candidate has been extracted from K, if the vessel cannot be assigned to that position and time because it would not have enough cranes throughout its processing, a new candidate consisting of the same position and the earliest berthing time in which there are enough cranes is included in K (Figure 5(a)). If the vessel at that time and position would overlap with one or more existing vessels in the space-time layout, the best candidate assignments in their contour are included in K (Figure 5(b)). Then, the next least-cost candidate in K is extracted and the process is repeated until a feasible assignment is found (Figure 5(c)). This is done for all the vessels in the list until the berth plan is completed.



(a) We try to assign vessel i at (a_i, d_i). Vessel i requires 2 cranes and the quay has only 3 cranes, so we look for a new assignment with enough cranes (empty circle).



(b) In the new assignment, vessel i would overlap with v_2, so we propose new candidates in the contour of v_2 (empty circles).



(c) Vessel i is assigned to the location of the least-cost feasible candidate.

Figure 5: Process of the Exploratory Constructive Algorithm for the BACAP.

4.3 Memetic improvement

Once the constructive algorithm has obtained a solution, we examine it looking for idle cranes, because it may be possible that the crane-to-vessel assignments do not require all Q cranes in each period. We go through the list of vessels, ordered by non-decreasing departure time, and if a crane is idle throughout the processing of a vessel, it is assigned to it. The reduction in processing time thus achieved can reduce or even eliminate the delay incurred by the vessel, and so its corresponding cost.

It is possible to attain a further improvement if we now feedback the constructive algorithm with the new list of cranes and the same list of vessels. The reduction in processing times previously obtained may now allow other vessels to be moored earlier. Therefore, the constructive algorithm is applied again and, if a better solution is obtained, the chromosome is updated with the refined list of cranes. This process is repeated until no further improvement is achieved. The memetic scheme thus performed is known in the literature as the *Lamarckian Memetic Algorithm*, since the local improvements are reflected in the chromosome and can therefore be inherited by the offspring (Le et al., 2009).

4.4 Local Search procedures

The Genetic Algorithm does not explore the entire solution space, but rather a subspace defined by the way in which the constructive algorithm generates feasible solutions following a list of vessels sequentially. In order to reach solutions that cannot be reached when allocating all the vessels one by one, we propose three different local search procedures based on directly moving vessels in an existing berth plan: two heuristics and one matheuristic. These LS procedures are used only as a final attempt to refine the best solutions attained by the BRKGA, so they are applied to all the individuals of each population as soon as the genetic process has ended. These individuals are sorted by non-increasing value, and starting from the best one, the procedure is applied until the stopping criterion, a time limit of $TimeLimitLS$ seconds, is met.

4.4.1 Simple ruin-and-recreate heuristic

The first heuristic is based on the ruin-and-recreate strategy, so we remove some vessels from the solution and then we use the constructive algorithm to reassign them.

This procedure first generates a list of vessels ordered by non-increasing score, which is, for each vessel, the sum of its cost and the cost of the vessels belonging to its cluster, which is determined by a neighbourhood function applied to the vessel. Then, a vessel is selected from the list with a probability $SelProb$, starting from the first element. The selected vessel and the vessels in its cluster are removed from the solution and reallocated in non-increasing cost order by means of the constructive algorithm, using the list of crane assignments of the original solution together with the refinement seen in Section 4.3, but without performing the feedback phase (Figure 6).

Additional solutions are generated by changing the number of cranes assigned to those vessels. We perform a *Path Relinking* on the list of cranes, keeping the list of vessels to be reassigned unchanged. The Path Relinking technique generates intermediate solutions in the trajectory between a *initiating solution* and a *guiding solution* by systematically changing elements of the initiating solution, step by step, until the guiding solution is reached. The initiating list is the original list of cranes, while the guiding list is selected between two alternatives: the list with the maximum number of cranes allowed for each of those vessels and the list with the minimum number of cranes. We consider that these candidates are ideal as guiding lists because they lead to extreme solutions. The constructive algorithm builds the solution corresponding to each list and the list leading to the best solution is selected.

Once a guiding solution has been chosen, the list of cranes at each step results from changing one crane-to-vessel assignment, replacing it with the number of cranes in the guiding list. Hence, a new solution is obtained for each new list of cranes. We generate a number $Paths$ of different paths between the initiating and the guiding lists. For the first path, the sequence of indexes that must be changed corresponds to the order in the list of vessels, while for the rest it is determined randomly.

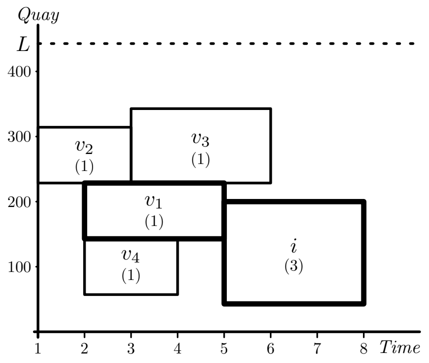
We store the best solution obtained, and if it is not better than the original one, the whole process is repeated with the next vessel in the list of vessels ordered by score. The number of tries is limited by a parameter $MaxTries$. Otherwise, if the solution is better, the whole process is repeated on the new best solution until no further improvement is achieved or until the stopping criterion is met.

We defined the following neighbourhood functions, which lead to different variants of the procedure:

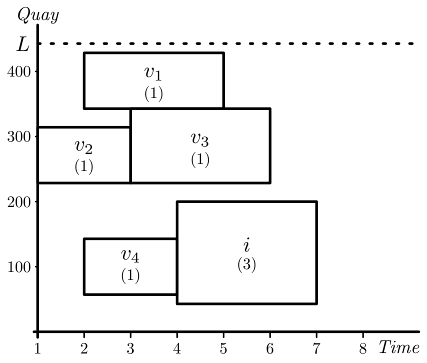
- *Overlapping vessels.* Given a vessel v, this function returns the vessels that would overlap with v if it were moved to its ideal location, that is, the one consisting of its desired position on the quay and its expected arrival time.

- *Adjacent vessels.* Given a vessel v, it returns the vessels adjacent to v: the vessels whose rectangles are touching the rectangle representing v in the graphic representation of the solution.

- *Connected component of vessel.* Given a vessel v, it returns the vessels adjacent to v and, recursively, all the vessels that are adjacent to them. In other words: if we consider the vessels as the vertices of a graph, connected by edges representing adjacency relations, the function returns the connected component to which v belongs.



(a) Vessel i is removed jointly with its adjacent vessel v_1. The quay has 5 cranes.

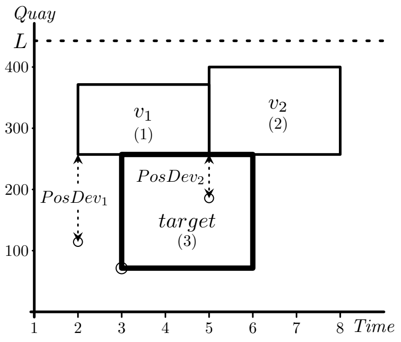


(b) Vessels i and v_1 are reassigned and several configurations of cranes are tested. This is a possible output.

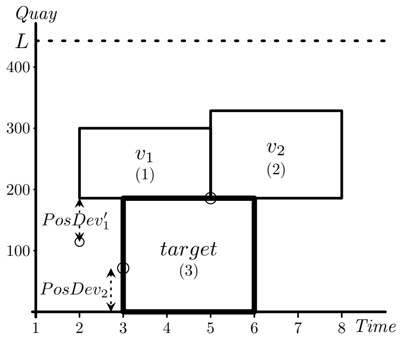
Figure 6: Simple ruin-and-recreate Local Search using the adjacency neighbourhood function.

4.4.2 Pushing ruin-and-recreate heuristic

This heuristic also removes a cluster of vessels, but before reassigning them, the ideal berthing time and position of a vessel is modified, so the constructive algorithm will try to put it at this new allocation. Looking at the solutions obtained by the Genetic Algorithm, we noticed a case in which the optimal solution cannot be reached due to the way in which the constructive algorithm builds the solutions. When assigning berthing times and positions to the vessels in a cluster, the first vessel was allocated with no cost, in its ideal assignment, at the expense of increasing the cost of all the other vessels. The optimal solution in those cases was reached by moving that vessel to a new position and/or time, increasing its cost, but allowing the other vessels to be better allocated so that the overall cost of the cluster was reduced. The best way to achieve this cluster reassignment would be to apply a mathematical programming model to the cluster, as we do in Section 4.4.3; however, the model can only be applied to small clusters. An alternative is to *push* the first vessel to a new location and reallocate the remaining vessels using the constructive algorithm. This requires determining heuristically the extent of the movement applied to that vessel, so we test different values calculated by considering the other vessels' deviations from their ideal assignments. In particular, we calculate maximum, minimum and average deviations in both time and position, so that we can try small and large displacements from the current location of the vessel. Figures 7 and 8 illustrate this process. In Figure 7 the target vessel is moved down slightly and the relative positions of the vessels in the cluster are not changed. In Figure 8 the target vessel is moved to a later time, allowing other vessels to be moored before it.

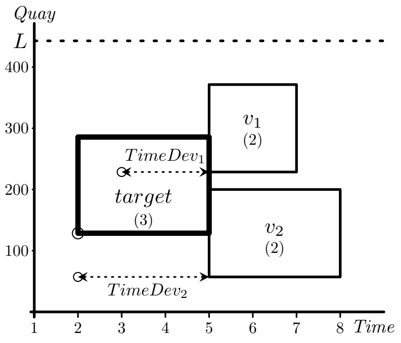


(a) The target vessel is in its ideal assignment. Vessels v_1 and v_2 are assigned to their arrival time, but deviate from their desired positions.

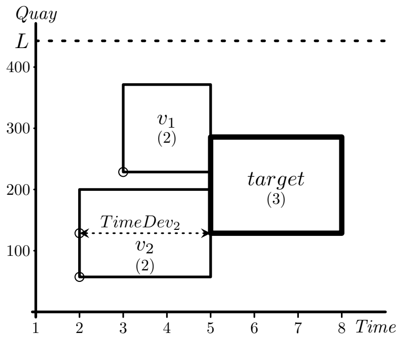


(b) The movement $(0, minPosDev)$ is applied to $target$. The other vessels are reassigned in order of non-increasing cost. Thus the overall cost is reduced.

Figure 7: Pushing ruin-and-recreate Local Search with the clustering neighbourhood function. Example of a position movement applied to the target vessel.



(a) The target vessel is in its ideal assignment. Vessels v_1 and v_2 are assigned to their desired positions on the quay, but deviate from their arrival time.



(b) The movement $(maxTimeDev, 0)$ is applied to $target$. The other vessels are reassigned in order of non-increasing cost. Thus the overall cost is reduced.

Figure 8: Pushing ruin-and-recreate Local Search with the clustering neighbourhood function. Example of a time movement applied to the target vessel.

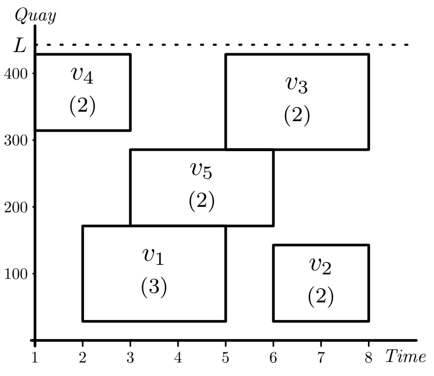
4.4.3 Matheuristic

An alternative method for improving the cluster reallocation is to use a mathematical programming model. In this case, we use a modified version of the mixed integer linear model that we developed for this problem (Correcher et al., 2017), which is summarized in Appendix A.

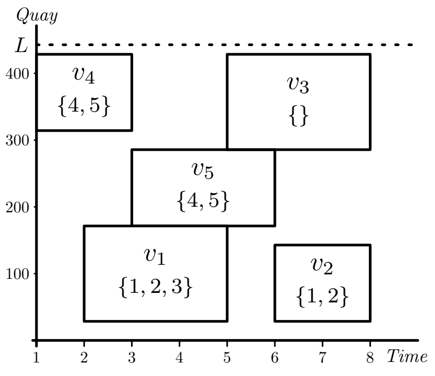
The initial steps of the procedure are similar to those of the previous one, but a cluster is selected only if its size is lower than or equal to a parameter *MaxClusterSize*. The cluster is completely removed and the model is used to reallocate the vessels in the cluster. We generate the model for the problem by inserting additional constraints to keep the values of berthing time and position variables fixed for the vessels not in the cluster. The only variables that the solver can change are those corresponding to the vessels in the cluster, so the size of the problem to be solved depends on the cluster size. This makes the method very useful for small clusters, given that a model with few vessels can be rapidly solved to optimality, as reported in the above-mentioned paper. Since vessels are not able to berth after the time horizon in the model, we construct it considering a planning horizon equal to the maximum of the original planning horizon and the maximum berthing time among the vessels in the solution: $H' = \max\{H, \max_{i \in V}(t_i)\}$. However, the original H is used to calculate the cost of exceeding the horizon. In order to reduce the search time, we also provide the model with the upper bound value of the current solution.

5 Extending the algorithms to address the BACASP

The Berth Allocation and Specific Quay Crane Assignment Problem (BACASP) is a BACAP in which the set of specific cranes that serve each vessel also has to be determined. This problem is important in the case of the time-invariant BACAP, given that a solution for this problem can be infeasible for the time-invariant BACASP. The difficulties arise when we face situations like the one depicted in Figure 9, where we consider 5 cranes numbered from 1 to 5 from the origin of the quay. It is impossible to assign the number of cranes determined by the BACAP solution to vessel v_3 without changing the specific cranes assigned to vessel v_5 during its processing. Alternatively, if we assigned cranes 4 and 5 to v_3 and cranes 2 and 3 to v_5, we would not be able to assign all the required cranes to v_1.



(a) BACAP solution for a terminal with 5 quay cranes. The number of cranes assigned is shown in parentheses.



(b) Attempting to generate a BACASP solution from the BACAP solution. The set of cranes assigned to each vessel is shown. It is impossible to assign cranes to v_3, as cranes 1–3 cannot cross working cranes 4 and 5.

Figure 9: A BACAP solution not feasible for the BACASP.

However, under certain conditions it is possible to obtain the optimal solution of the BACASP from the optimal solution of the corresponding BACAP in polynomial time. Türkoğullari et al. (2014) defined a vessel sequence $v_1, v_2, \ldots, v_n$ in a given feasible solution of BACAP to be *complete* if v_1 is the vessel closest to the beginning of the quay, v_n is the closest to the end of the quay, v_i and v_{i+1} are two consecutive vessels with v_i closer to the beginning of the quay, and each pair of consecutive vessels in the sequence is concurrent during at least one time period. A complete sequence is said to be *proper* when the sum of the number of cranes assigned to the vessels in this sequence is less than or equal to the number of cranes at the quay. Otherwise, it is called an *improper* complete sequence. For example, in Figure 9(b) vessels v_1, v_5, and v_3 form a improper complete sequence. Using these definitions, Türkoğullari et al. (2014) proved that, given an optimal solution $\boldsymbol{X^*}$ of the BACAP, an optimal solution of the BACASP can be obtained from it if and only if every complete sequence of vessels extracted from $\boldsymbol{X^*}$ is proper. They also provided a polynomial-time algorithm which is useful when the condition is satisfied, whereas for cases where it is not, they proposed a cutting plane algorithm based on their BACAP model.

We use these results to adapt our algorithms for the BACASP case. This is done by changing the constructive algorithm so that it always generates solutions not containing improper complete sequences. Thus, the proposed Local Search procedures based on the constructive algorithm do not need additional adjustments. By contrast, the matheuristic Local Search now applies the cutting plane algorithm based on our BACAP model (Correcher et al., 2017). The BACAP solutions thus obtained are easily transformed into BACASP solutions by applying the polynomial-time algorithm (Türkoğullari et al., 2014).

The constructive algorithm is the same as that proposed in Section 4.2 for the BACAP, the only difference being that it now has to prevent vessels being assigned to locations in which they would generate improper complete sequences. Thus, when trying to assign berthing position and time to a vessel i, after checking that the candidate assignment satisfies the number of cranes required and does not overlap with any other vessel, we also check whether it creates one or more improper complete sequences. If so, new minimum-cost alternative candidate assignments that break all the improper complete sequences have to be added. To prevent falling into cycles of repeated candidates, we include a set of already visited candidates. Figure 10 illustrates this process for the case in which vessel i would create an improper complete sequence when assigned to its ideal location (a_i, d_i). The constructive algorithm continues until each vessel is assigned to a position and a time in which all constraints are satisfied.

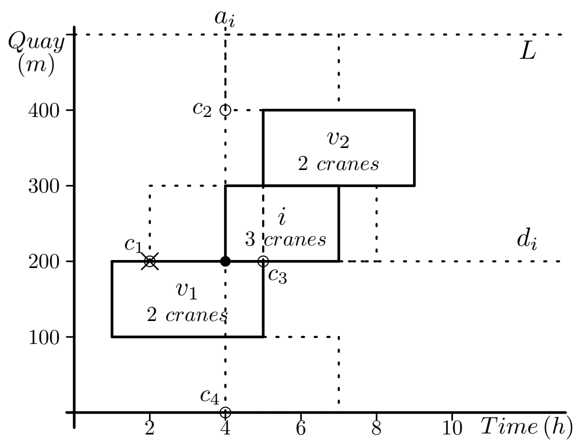


Figure 10: Example of applying the BACASP constructive algorithm to a quay with 5 cranes. Vessel i generates an improper complete sequence (v_1, i, v_2) in its ideal assignment. Alternative assignments $c_1 \dots c_4$ are considered in order to break the sequence. Note that candidate c_1 is immediately discarded because the time is lower than the arrival time a_i.

6 Computational experiments

We conducted several computational experiments to evaluate the quality of our approach. All of them were run on an Intel Core i7 2600 at 3.4 GHz with 31.4 GiB of RAM. We used C++11 as the programming language and the OpenMP parallel programming library. We evaluated as many individuals as possible in parallel threads, both in the Genetic Algorithm and in the Local Search procedures. To solve the mathematical programming model, we used CPLEX 12.6, limiting the size of the search tree to 30 GiB. The statistical analysis was performed using the programming language R 3.3 and package *ez* 4.4.

Several sets of instances were generated according to different criteria used in previous papers. The first set of instances is inspired by the criteria applied by Meisel and Bierwirth (2009). Henceforth, it will be referred to as *InstMB*. It consists of 90 instances: 10 for each number of vessels considered, from 20 to 100. The time unit is 1 hour and the space unit is 10 meters. The quay is 1000 meters long and is equipped with 10 cranes. There are three different kinds of vessels: Feeder, Medium and Jumbo. For each instance, 60%, 30% and 10% of the vessels correspond to these classes respectively. The arrival time of the vessels is uniformly distributed in $[0, 168]$, and the time horizon is 210 hours in instances of up to 70 vessels and 252 hours in those with 80–100 vessels. The lengths, workloads, and min/max number of cranes of the vessels are computed according to Table 1, while the desired position of each vessel i is generated from $U[1, L + 1 - l_i]$, rounding up to the nearest integer. For each vessel, its processing time for a given number of cranes assigned to it results from dividing its workload in crane-hours by the number of cranes, rounding up to the nearest integer. The desired departure time of each vessel i is $1.5 \cdot (a_i + u_i^{q_i^{max}})$. The costs are the same for all the vessels, so none of them is privileged over the others: $C_i^w = 1000$, $C_i^d = 2000$, $C_i^p = 200$ and $C_i^h = 5000$.

The second set was generated applying the criteria of Park and Kim (2003). From now on, it will be referred to as *InstPK*. The quay is 1200 meters long and the planning horizon is 300 hours discretized in units of 10 m and 1 hour respectively. There are 11 quay cranes available and vessels can only use from 2 to 5 cranes. The set consists of 50 randomly generated instances, 10 instances for each number of vessels considered: $N \in \{20, 25, 30, 35, 40\}$. The data relating to each vessel are determined by uniform distributions as follows: $U[1, 170]$ for the arrival time, $U[10, 48]$ for the number of crane-hours required, $U[15, 35]$ for the length, and $U[1, 120]$ for the desired position on the quay. The processing time of each vessel results from dividing the number of single-crane-hours by the number of cranes, rounding up to the nearest integer. The desired departure time, which is not specified in their paper, is determined by applying the criterion of Meisel and Bierwirth (2009): $s_i = 1.5 \cdot (a_i + u_i^{q_i^{max}})$. Finally, the costs are the same for all the vessels: $C_i^w = 1000$, $C_i^d = 2000$, $C_i^p = 200$ and $C_i^h = 5000$.

Table 1: Specifications for the different classes of vessels in set *InstMB*.

Class	l_i (m)	*Workload* (single-crane-hours)	q_i^{min}	q_i^{max}
Feeder	$U[8, 21]$	$U[5, 15]$	1	2
Medium	$U[21, 30]$	$U[15, 50]$	2	4
Jumbo	$U[30, 40]$	$U[50, 65]$	4	6

We conducted four different experiments on BACAP algorithms and afterwards they were repeated for their BACASP versions. In the first one we looked for a good configuration of the parameters of the Genetic Algorithm without performing Local Search (Section 6.1). In the second, first we tried to determine a good parameter configuration for each LS algorithm and then we compared them to select the best one (Section 6.2). In the third experiment, we evaluated the performance of the entire algorithm comparing its results with the best solutions achieved by the MILP (Section 6.3). Finally, the forth experiment compared the proposed Genetic Algorithm with an implementation of the Discrete Differential Evolution metaheuristic (Section 6.4).

In order to perform a correct adjustment of the algorithms we generated a training set of instances, called $InstAdjust$, applying the same criteria as in the set $InstMB$, since they are, to our knowledge, more realistic than those of $InstPK$. The set $IntAdjust$ consists of 45 instances in total: 5 for each number of vessels considered, from 20 to 100.

6.1 Parameter adjustment of the Genetic Algorithm

The objective of the first experiment was to determine the best values of the parameters of the Genetic Algorithm. The parameters $E = 15\%$, $I = 15\%$, and $CrossBias = 0.7$ were fixed, since we considered that these were reasonable values that produced good results, according to previous studies (Gonçalves and Resende, 2012) and our preliminary experiments. In these preliminary experiments we also noticed that the most influential parameters seemed to be: $MultPop$, N_{pop}, and $GensMig$. Hence we decided to test different values for them: $MultPop \in \{1, 10, 20\}$, $N_{pop} \in \{1, 10, 20\}$, and $GensMig \in \{10, 30\}$. This resulted in a total of 15 combinations. The time limit was set to $TimeLimitGen = N$ seconds, so that the computation time for each instance depended on its size.

For each parameter configuration we ran five independent random repetitions of the BRKGA on each instance in $InstAdjust$. For each repetition we recorded the value of the best solution achieved and the running time. We then calculated the deviation percentage (DFB) of the result obtained in each repetition ($value$) from the minimum value attained among the repetitions of all the configurations tested on the same instance ($minValue$): $DFB = 100 \cdot (value - minValue)/minValue$.

In order to determine whether the differences in DFB were statistically significant, we conducted a *repeated measures design*, taking the *average* DFB obtained for each instance from its five repetitions as the dependent variable. Since we ran each configuration over the same instances, we considered the *instances* as the subjects of the analysis, while $configuration$ was considered a within-subjects fixed factor with 15 levels corresponding to the different configurations studied. Thus, the 15 DFB means obtained for each instance were considered different measures on the same subject. Moreover, as the performance of the algorithm may be different for different problem sizes, we considered the *number of vessels* in the instance as a between-subjects fixed factor, with 9 levels (20–100 vessels). Including these factors allows the statistical analysis to manage the variability they introduce, thereby increasing its sensitivity. We established a significance level of $\alpha = 0.05$ for all the statistical tests. The assumptions of validity of this analysis (Maxwell and Delaney, 2004) were tested on each experiment and considered acceptable.

6.1.1 Adjustment of the BACAP Genetic Algorithm

The ANOVA tested the effect of *configuration*, the effect of the *number of vessels* and the interaction between them. All these effects proved to be statistically significant, with $p < 0.0001$ in all of them. The effect that explained the largest proportion of the sample DFB variability was the *number of vessels*. This indicates that regardless of the algorithm configuration applied, the size of the instance has a strong impact on the performance. Such evidence is consistent with our expectations, given that heuristics also suffer the effects of the combinatorial explosion, although they can withstand it better than solvers implementing mathematical programming models.

Given that we were interested in the *configuration* effect to determine which is the best regardless of the size of the instance and within the margins of our study, we conducted the Holm-Bonferroni multiple comparison tests for this effect alone, specifying that they were paired samples. we thereby obtained the groups of configurations within which their average DFB cannot be considered statistically different. Table 2 shows the average number of generations done and the mean and the standard deviation of DFB for each configuration, together with the homogeneous groups. According to the results, configurations with the highest total number of individuals ($N \cdot MultPop \cdot N_{pop}$) show a performance clearly worse than the rest. Examining the data we observed that in those configurations, the DFB in instances with more than 50 vessels was higher than in any other configuration. This is due to the small number of generations performed. Indeed, the configurations that obtained the best DFB results completed more than 24 times the number of generations completed by the worst. Therefore, performing more generations is preferable to increasing the number of populations and individuals therein. Apart from this, the $GensMig$ values tested do not seem to make a significant and consistent difference. The configurations of group B are those with the lowest means, so for the rest of the study we chose the one with the lowest sample average among them: $MultiPop = 1$, $N_{pop} = 20$, and $GensMig = 30$.

Table 2: Results of the BACAP Genetic Algorithm on the set *InstAjust*.

Configuration			Generations	*DFB* (%)		Group		
$MultPop$	N_{pop}	$GensMig$	Mean	Mean	Std. Dev.	A	B	C
1	1	10	30612.8	7.29	6.60	×		
1	10	10	3177.3	5.18	4.30		×	
1	10	30	3180.4	4.96	4.41		×	
1	20	10	1576.1	4.76	4.17		×	
1	20	30	1585.7	4.44	3.89		×	
10	1	10	3017.0	5.89	6.22	×	×	
10	10	10	283.9	5.10	5.32		×	
10	10	30	284.6	5.33	5.82		×	
10	20	10	139.3	8.36	8.48	×		
10	20	30	140.8	8.28	8.45	×		
20	1	10	1420.5	5.77	6.47	×	×	
20	10	10	130.7	8.99	8.92	×		
20	10	30	131.8	8.99	8.87	×		
20	20	10	64.4	14.30	13.90			×
20	20	30	64.7	14.18	13.68			×

6.1.2 Adjustment of the BACASP Genetic Algorithm

The same experiment was conducted on the BACASP Genetic Algorithm and the results were studied with the same type of ANOVA. The tests showed statistically significant differences in the average DFB due to the interaction between *number of vessels* and *configuration*, *configuration* alone, and *number of vessels* alone, with $p < 0.0001$ in all of them. These values are similar to those obtained with the BACAP version, so the same conclusions can be drawn from them. In order to assess the differences between the configurations, we applied the Holm-Bonferroni multiple comparisons again, specifying that they were paired samples. Table 3 shows the results, in which we can appreciate the same effect of the total number of individuals on the number of generations completed and the consequent differences in performance. Compared with the BACAP Genetic Algorithm, this algorithm performs approximately half the number of generations, which clearly reflects the complexity of the BACASP and the additional time spent on generating solutions without improper complete sequences. Group B consists of the best configurations, from which we selected the one with the lowest sample average for the rest of the study: $MultiPop = 10$, $N_{pop} = 1$, and $GensMig = 10$.

Table 3: Results of the BACASP Genetic Algorithm on the set *InstAjust*.

Configuration			Generations	DFB (%)		Group			
$MultPop$	N_{pop}	$GensMig$	Mean	Mean	Std. Dev.	A	B	C	D
1	1	10	13768.1	9.56	8.21	×			
1	10	10	1375.6	6.28	4.99		×		
1	10	30	1377.5	6.71	5.51		×		
1	20	10	684.8	5.91	5.02		×		
1	20	30	685.8	6.94	5.79		×		
10	1	10	1511.0	5.16	5.04		×		
10	10	10	141.4	13.38	13.77	×			
10	10	30	141.3	13.71	14.08	×			
10	20	10	68.8	21.59	20.36			×	
10	20	30	68.8	21.70	20.36			×	
20	1	10	694.5	5.42	4.98		×		
20	10	10	63.5	22.40	20.81			×	
20	10	30	63.6	22.70	20.58			×	
20	20	10	30.7	33.72	29.82				×
20	20	30	30.8	33.58	29.46				×

6.2 Adjustment of the Local Search procedures

After determining a proper configuration for each Genetic Algorithm, we conducted analogous experiments to adjust the parameters of the local search procedures. In particular, we evaluated the following algorithms: the simple ruin-and-recreate heuristic with the overlap neighbourhood function (*SimpleOverlap*), with the adjacency function (*SimpleAdjacency*), and with the connected component function (*SimpleConnected*); the pushing ruin-and-recreate heuristic with the connected component function (*Pushing*); and the matheuristic with the adjacency function (*ModelAdjacency*) and with the connected component function (*ModelConnected*). The pushing heuristic was used only with the connected component function due to its very design, which is tailored to it. As for the matheuristic, the overlap function was not considered because it performed worse in preliminary experiments.

We set an overall time limit of one second per vessel in each instance (N), of which $TimeLimitGen = 0.9N$ and $TimeLimitLS = 0.1N$ seconds. In preliminary experiments we noticed that all the local search algorithms except the matheuristic were able to finish within this time limit considering $MaxTries = N$, hence we fixed that value for them. For the other parameters of the algorithms we tried all the combinations: $SelProb \in \{0.5, 0.75, 1\}$ and $Paths \in \{30, 45, 60\}$. For the matheuristic we fixed $MaxClusterSize = 15$, as we observed that greater values made it so slow that it could not solve many models, and we tried all the combinations of $SelProb \in \{0.5, 0.75, 1\}$ and $MaxTries \in \{1, 5, 10\}$. This results in a total of 9 different configurations for each LS procedure.

A single repetition of the experiment consisted of running the Genetic Algorithm on each instance and applying each LS algorithm with each configuration independently to its result. In this way we could properly compare their performance, since all of them were applied to the set of individuals resulting from the same run of the Genetic Algorithm.

6.2.1 Adjustment and comparison of the Local Search procedures for the BACAP

First we performed six separate ANOVAs, one for each Local Search procedure, in order to study the effect of the parameters, to identify homogeneous groups, and finally to determine the best configuration of each algorithm. Then we did an ANOVA to compare the six LS algorithms and the Holm-Bonferroni multiple comparisons over the *algorithm* factor in order to identify groups of algorithms with non-significantly different averages of DFB.

The results of the ANOVA for the BACAP Local Search algorithms showed that the interaction of *number of vessels* and *algorithm*, *algorithm* alone, and *number of vessels* alone produced statistically significant differences between the means, with p values less than 0.001 in all of them. Table 4 shows the best configuration of each algorithm in columns 2-4 and the mean and standard deviation of DFB as well as the homogeneous groups in the remaining columns. One group clearly stands out: group C, with the lowest mean values of DFB, corresponding to the two matheuristics, so we can conclude that matheuristics clearly outperform the other Local Search procedures. Therefore, out of all the algorithms we selected *ModelAdjacency*, as it belongs to the best homogeneous group and its sample mean is slightly lower than that obtained by $ModelConnected$.

Table 4: Results of the multiple comparisons between the LS algorithms for the BACAP.

Algorithm	Configuration			DFB (%)		Group		
	$MaxTries$	$SelProb$	$Paths$	Mean	Std. Dev.	A	B	C
$SimpleOverlap$	N	1	60	5.36	3.40	×		
$SimpleAdjacency$	N	1	60	4.97	3.32		×	
$SimpleConnected$	N	0.75	30	5.43	3.44	×		
$Pushing$	N	0.5	30	5.21	3.58	×	×	
$ModelAdjacency$	10	1	-	2.88	2.06			×
$ModelConnected$	10	1	-	3.35	2.94			×

The selected Local Search procedure shows a performance better than the others, but it is important to analyse whether it is better than using the Genetic Algorithm without any LS algorithm applied at the end. This was analysed through an ANOVA similar to the previous ones. The ANOVA revealed that *number of vessels*, *algorithm*, and their interaction were all significant effects, with p values less than 0.0001. Taking into account that the sample mean DFB obtained by the Genetic Algorithm alone was 5.45% and the one obtained by *ModelAdjacency* was 2.88%, we can conclude that there is a clear performance gain when using this Local Search algorithm.

6.2.2 Comparison between the BACASP Local Search procedures

The results for the BACASP, applying the same analysis as in the previous case, are shown in Table 5. In this case, the algorithm *ModelAdjacency*, the only element in group C, shows the lowest mean. Therefore, we selected this algorithm for the final version of the complete algorithm.

Table 5: Results of the multiple comparisons between the LS algorithms for BACASP.

Algorithm	Configuration			DFB (%)		Group		
	$MaxTries$	$SelProb$	$Paths$	Mean	Std. Dev.	A	B	C
$SimpleOverlap$	N	1	60	6.18	4.81	×		
$SimpleAdjacency$	N	1	60	5.71	4.66		×	
$SimpleConnected$	N	0.75	30	6.36	4.81	×		
$Pushing$	N	0.75	30	6.28	4.85	×		
$ModelAdjacency$	10	1	-	3.51	3.11			×
$ModelConnected$	10	1	-	4.95	4.31		×	

In order to assess whether the Local Search improves significantly on the results obtained by the Genetic Algorithm alone, we conducted an analogous ANOVA. The analysis showed that *number of vessels*, *algorithm*, and their interaction were all statistically significant, with p values less than 0.0001. The sample mean DFB of the Genetic Algorithm was 6.39%, and the one for *ModelAdjacency* was 3.51%, which is significantly lower. Consequently, we conclude that the selected LS procedure is worth applying after the BRKGA, as it significantly improves on its results.

6.3 Evaluation of the Genetic Algorithm with the best Local Search

The selected configuration of the Genetic Algorithm with LS was evaluated over the sets of instances $InstMB$ and $InstPK$. In this section, its results are compared with those achieved by CPLEX 12.6 solving the integer linear model reported in Correcher et al. (2017) with a time limit of 3600 seconds. In the next section, the evaluation is completed comparing the proposed algorithm with an implementation of the Discrete Differential Evolution algorithm.

We performed five random repetitions of the Genetic Algorithm on each instance in both sets. As we were interested in obtaining results in short times, the overall time limit was set to N seconds (one second per vessel), dedicating 90% of the time to the Genetic Algorithm itself and 10% to the Local Search. For each instance and repetition we recorded the cost of the best solution achieved by the Genetic Algorithm alone (*resultGen*) and the one attained after applying the LS procedure (*resultGenLS*). Then we obtained the minimum result of the five repetitions (*minResultGenLS*) and the deviation percentage of this result from the result of the integer model (*resultModel*) for the same instance: $minDFM =$

$(minResultGenLS - resultModel)/resultModel$. We also calculated the average $resultGenLS$ over the five repetitions ($avgResultGenLS$) and the deviation percentage: $avgDFM = (avgResultGenLS - resultModel)/resultModel$. The average $resultGen$ over the five repetitions ($avgResultGen$) was computed as well to obtain the mean improvement achieved by the LS: $ILS = (avgResultGen - avgResultGenLS)/avgResultGenLS$. In order to assess the performance of this approach with other reasonable time limits, we repeated the experiment for $3N$, keeping the same time proportions for the *BRKGA* and the *LS*. Even for the larger instances of 100 vessels, these results are obtained in 5 minutes, which seems acceptable in practical situations.

6.3.1 Evaluation of the BACAP Genetic Algorithm

The mixed integer linear model was solved to optimality for all the instances in $InstMB$ with up to 40 vessels and seven of those containing 50 vessels. In two of the instances with 70 vessels, the model could not even obtain a feasible solution, and this also occurred in all the instances containing more than 70 vessels. In one instance of 40 vessels in $InstPK$, optimality was not proven.

The results of the BACAP Genetic Algorithm are summarised in Table 6. For each instance size, it shows the mean running time of the MILP; the mean and maximum $minDFM$ and $avgDFM$, considering only the instances for which both the MILP and the algorithm obtained feasible solutions; the percentage of instances solved to optimality by the BRKGA in at least one repetition relative to the number solved optimally by the model, and the percentage of instances solved optimally in all the repetitions; the mean ILS; and the mean cost. The missing values indicate that it was impossible to compare with the model because it could not achieve even feasible solutions. In the case of the number of optimums, it indicates that it is impossible to know whether any optimum was reached, as the model did not achieved any optimal solution.

It can be observed in column 2 that the integer model works well on small instances and can be considered a good solution strategy when the number of vessels does not exceed 60, although with sharply increasing computing times. In fact, 60 vessels seems to be the limit for the MILP. For instances with 70 vessels, no solution was found for some instances, and when a feasible solution was found, it was not very good, as compared with the solution obtained by the Genetic Algorithm. The remaining columns in Table 6 show the performance of the BRKGA. On very short computing times (1 second per vessel), it is able to obtain solutions quite close to the optimal solutions, with average distance no greater than 5% if we consider the best solution obtained in the 5 independent runs on each instance and no greater than 9% if we consider the average value of the 5 runs, although columns 4 and 6 indicate that there are some instances for which the Genetic Algorithm fails to obtain a really good result. In addition, the contribution of the LS, shown in column 9, is notable, reaching a mean of 2.29% in $InstMB$ and 0.59% in $InstPK$, complementing the solution space search done by the Genetic Algorithm in a very efficient way. The last column shows the average cost of the solutions. The sharply increasing values indicate that if a large number of vessels have to be allocated within a planning horizon of one week, the problem is highly complex and many conflicts between vessels have to be solved.

Table 6: Results of the BACAP algorithm on *InstMB* and *InstPK* with an overall time limit of one second per vessel (N).

N	Mean time	$minDFM$ (%)		$avgDFM$ (%)		Opt. (%)		Mean	Mean
	MILP model (s)	Avg.	Max.	Avg.	Max.	One	All	ILS (%)	cost
InstMB									
20	0.6	0	0	0.22	2.18	100	90	3.65	13224
30	4.5	0.97	9.36	1.93	9.36	80	30	2.63	41812
40	9.5	1.89	4.43	3.47	10.39	30	20	3.29	55632
50	2084.4	5.00	13.23	8.04	17.20	0	0	1.97	157796
60	3602.3	4.36	13.21	7.32	17.58	-	-	2.48	304000
70	3601.8	-19.05	10.69	-15.78	15.02	-	-	1.90	709388
80	-	-	-	-	-	-	-	2.21	1226960
90	-	-	-	-	-	-	-	1.36	3108228
100	-	-	-	-	-	-	-	1.09	4652508
InstPK									
20	1.2	0.57	5.73	1.21	5.73	90	70	0.58	22300
25	1.7	1.01	10.05	1.21	12.06	90	90	0.05	31936
30	4.0	1.09	5.06	2.04	5.56	60	40	0.55	51224
35	53.8	3.19	10.23	4.92	12.92	40	30	0.40	85676
40	595.2	4.99	10.86	8.62	20.84	22	11	1.38	146680

The results obtained considering a time limit of $3N$ seconds per instance are shown in Table 7. The Genetic Algorithm and the associated Local Search make good use of this extended computing time. Minimum, average, and maximum distances to optimal solutions decrease significantly and the number of optimal solutions found by the metaheuristic increases. The average cost of instances in *InstMB* and *InstPK* is reduced by 2.29% and 1.12% respectively.

Table 7: Results of the BACAP algorithm on $InstMB$ and $InstPK$ with an overall time limit of three seconds per vessel ($3N$).

N	Mean time	$minDFM$ (%)		$avgDFM$ (%)		Opt. (%)		Mean	Mean
	MILP model (s)	Avg.	Max.	Avg.	Max.	One	All	ILS (%)	cost
$InstMB$									
20	0.6	0	0	0	0	100	100	3.89	13200
30	4.5	0.94	9.36	1.61	9.36	90	40	3.03	41716
40	9.5	1.78	4.35	2.39	6.43	30	30	3.70	55104
50	2084.4	3.45	10.49	6.20	15.21	14	0	2.53	154760
60	3602.3	1.59	6.38	5.85	11.66	-	-	2.39	299488
70	3601.8	-21.23	5.42	-17.65	12.11	-	-	2.20	703168
80	-	-	-	-	-	-	-	2.39	1196556
90	-	-	-	-	-	-	-	1.44	3049068
100	-	-	-	-	-	-	-	1.22	4526332
$InstPK$									
20	1.2	0.57	5.73	0.94	5.73	90	70	0.83	22252
25	1.7	1.01	10.05	1.14	11.43	90	90	0.04	31912
30	4.0	0.96	5.06	1.92	5.75	70	40	0.53	51176
35	53.8	2.69	9.65	3.64	9.65	50	30	0.48	84584
40	595.2	3.43	8.14	7.14	14.42	22	0	1.47	144140

In summary, these experiments show that the BACAP Genetic Algorithm accomplishes the main objective of obtaining good solutions to both small and large instances in short times. The percentage of optimal solutions attained indicates that it performs a good exploration of the solution space. Indeed, the mean deviations show that it is the best option for instances with more than 60 vessels and a fast solving method for the rest. The selection of a time limit above $3N$ makes it possible to improve on the results, but the Genetic Algorithm seems close to convergence in most of the instances, so the improvement is expected to decrease as the time is increased.

6.3.2 Evaluation of the BACASP Genetic Algorithm

The BACASP Genetic Algorithm was compared with the cutting plane algorithm for the BACASP (CPA) proposed by Correcher et al. (2017), implemented in C++ and CPLEX 12.6 and run for 3600 seconds (Appendix B) considering $MaxSizeSeqForPerm = 4$. By using this method, it was possible to achieve optimal solutions in all the instances with up to 40 vessels and in two instances with 50 vessels in $InstMB$. In $InstPK$, only one instance with 35 vessels and three with 40 vessels could not be solved to optimality.

In Table 8 we show the results obtained by the Genetic Algorithm in both sets of instances. In general, they are similar to those obtained for the BACAP, except for a slight decrease in the size of the instances that CPA was able to solve. In fact, in 8 instances with 50 vessels and in all the instances with more vessels no feasible solution was found. It can also be observed that the Genetic Algorithm did not perform well on at least two instances, one with 20 vessels in $InstMB$ and the other with 40 vessels in $InstPK$, whose DFM exceeded 40%. These values affected the means of their corresponding groups, hence the high

$avgDFM$ observed in the instances with 40 vessels in $InstPK$. Nevertheless, it is worth noting that the BACASP is more difficult to solve than the BACAP and the Genetic Algorithm provides a good alternative for larger instances. As for the Local Search, it attained a mean improvement of 1.58%, proving again to be a good complement to the Genetic Algorithm.

Table 8: Results of the BACASP algorithm on *InstMB* and *InstPK* with an overall time limit of one second per vessel (N).

N	Mean time	$minDFM$ (%)		$avgDFM$ (%)		Opt. (%)		Mean	Mean
	MILP model (s)	Avg.	Max.	Avg.	Max.	One	All	ILS (%)	cost
$InstMB$									
20	0.7	5.14	40.48	5.14	40.48	80	80	0.18	14100
30	13.5	2.53	9.76	2.70	9.76	40	30	3.13	43316
40	85.8	2.24	4.80	4.44	12.44	20	10	3.15	59276
50	3292.1	4.06	6.43	6.51	8.80	0	0	1.29	177960
60	3604.5	-	-	-	-	-	-	1.62	363836
70	3603.1	-	-	-	-	-	-	1.92	838164
80	-	-	-	-	-	-	-	1.49	1438184
90	-	-	-	-	-	-	-	0.82	3311732
100	-	-	-	-	-	-	-	0.62	4778248
$InstPK$									
20	1.8	0.97	8.33	1.14	8.33	80	70	0.00	22996
25	3.1	0.46	2.30	0.76	2.58	80	60	0.01	32240
30	15.4	0.17	1.11	2.14	6.33	80	50	0.44	53808
35	640.6	1.75	5.34	5.23	12.07	44	22	0.39	90884
40	1516.2	12.3	36.67	20.10	53.86	0	0	3.35	170744

The experiment conducted considering a time limit of $3N$ yielded some improvements relative to the results obtained in N seconds, especially by reducing the maximum $avgDFM$ (Table 9). It also reached new optimal solutions in five instances in set $InstPK$ and reduced the mean cost by 1.85% in $InstMB$ and 2.68% in $InstPK$. Both experiments show that the proposed Genetic Algorithm is also a good approach for solving the BACASP, particularly when more than 40 vessels are expected to arrive within a time horizon of one week, as the exact method cannot even find feasible solutions.

Table 9: Results of the BACASP algorithm on *InstMB* and *InstPK* with an overall time limit of three seconds per vessel ($3N$).

N	Mean time	*minDFM* (%)		*avgDFM* (%)		Opt. (%)		Mean	Mean
	MILP model (s)	Avg.	Max.	Avg.	Max.	One	All	ILS (%)	cost
InstMB									
20	0.7	1.33	10.91	4.38	32.86	80	80	0.64	13972
30	13.5	2.45	9.76	2.68	9.76	30	30	3.01	43300
40	85.8	2.03	4.80	2.97	5.84	20	10	2.31	58272
50	3292.1	4.06	6.43	4.61	6.64	0	0	1.79	174568
60	3604.5	-	-	-	-	-	-	3.03	355704
70	3603.1	-	-	-	-	-	-	1.90	818300
80	-	-	-	-	-	-	-	1.74	1390468
90	-	-	-	-	-	-	-	1.02	3231884
100	-	-	-	-	-	-	-	0.90	4738120
InstPK									
20	1.8	0.97	8.33	0.97	8.33	80	80	0	22940
25	3.1	0	0	0.35	1.57	100	60	0.01	32068
30	15.4	0.53	1.94	1.65	6.96	70	60	0.32	53428
35	640.6	1.54	5.89	3.53	12.15	55	44	0.72	89516
40	1516.2	7.09	29.52	13.62	37.43	14	0	2.71	163052

6.4 Comparing the Genetic Algorithm with a Discrete Differential Evolution algorithm

In order to assess the performance of the proposed Genetic Algorithm on large instances for which the integer linear model did not obtain good solutions, we implemented a Discrete Differential Evolution (DDE) algorithm, which has proven to yield good results in scheduling problems. Differential Evolution was first proposed by Storn and Price (1997) as a population-based metaheuristic to solve continuous optimization problems. Later on it was adapted and applied by Tasgetiren et al. (2007a,b) and Pan et al. (2007) to tackle combinatorial optimization problems, leading to the Discrete Differential Evolution algorithm. This method also works with chromosomes representing diverse solutions to the problem, which are mutated and recombined over several generations. Unlike Genetic Algorithms, it focuses on the best individual achieved in the previous generation (referred to hereafter as π_g^{t-1}) in order to generate new individuals. In particular, the scheme of the DDE followed here is the same as in Pan et al. (2008), except that we do not apply an internal Local Search and the single initial population is generated exactly as in the case of the BRKGA proposed here.

A mutated individual results from modifying π_g^{t-1} with probability P_m. This process first randomly selects d vessels and sets a random number of cranes for each vessel out of its allowed number of cranes. Then, the Deconstruction-Construction algorithm (DC) referred to in the cited paper is applied to the list of vessels with parameter d, using the Exploratory Constructive Algorithm internally with the memetic improvement that we proposed. The *mutant population* thus consists of N_{indiv} individuals: the mutant individuals and the individuals not affected according to P_m, which are copies of π_g^{t-1}.

Next, the mutant population is paired with the population of the previous generation according to the index of each individual. Each of these pairs is recombined to generate a new individual with a probability P_c. The list of cranes of the child results from randomly selecting the list of cranes of one of the parents with probability 0.5. Then, two positions in the list are randomly selected and the numbers of cranes corresponding to the vessels within them (inclusive) are replaced with those in the same positions in the other parent. The list of vessels is created analogously, applying the PTL algorithm referred to in the cited paper. The resulting population of N_{indiv} individuals consists of these individuals and those not recombined according to P_c, which will be the individuals already in the mutant population. This population is called the *trial population* and is evaluated to determine the fitness of each individual.

Finally, the population of the next generation is created by pairing the trial population with the population of the previous generation and choosing, for each index, the least-cost individual in each pair. The stopping criterion for the iterative process is a time limit, after which the *ModelAdjacency* Local Search procedure with its best parameter configuration is applied to all the individuals in the last population until a time limit is reached, as in the case of the BRKGA.

The DDE algorithm was implemented and run on the same computer under the same conditions, using OpenMP to make the most of the processors in both the evaluation of the individuals and the DC mutation phase, which makes intensive use of the constructive algorithm. The parameters chosen were those used in the cited paper ($d = 4$, $P_m = 0.2$, and $P_c = 0.8$), the population size was $N_{indiv} = 10N$ and the overall time limit was N in the first experiment and $3N$ in the second, devoting 90% of the time to the DDE and 10% to the Local Search, as in the BRKGA. The DDE algorithm for the BACASP was implemented analogously, using the modified versions of the constructive algorithm and the *ModelAdjacency* Local Search procedure. The algorithms were run on the same sets of instances, performing 5 repetitions on each instance.

Tables 10 and 11 show for each group of instances the average and maximum percentage deviations of the results of the BRKGA from the results of the DDE, where $avgDFD$ is the deviation of the average result obtained by the BRKGA in the five runs from the average result obtained by the DDE algorithm in the five runs ($avgResultDDE$), calculated as $avgDFD = (avgResultGen - avgResultDDE)/avgResultDDE$. The deviation obtained from the results of the algorithms applying the Local Search procedure as well ($avgDFD_{LS}$) is calculated analogously.

Table 10: Comparison of the BRKGA with the DDE algorithm for the BACAP on $InstMB$ and $InstPK$, considering N and $3N$ seconds as time limits

	N seconds				$3N$ seconds			
	$avgDFD$ (%)		$avgDFD_{LS}$ (%)		$avgDFD$ (%)		$avgDFD_{LS}$ (%)	
Vessels	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
$InstMB$								
20	-0.83	0	0	0	-0.22	0	0	0
30	-1.91	0	-1.85	1.27	-1.42	0	-2.12	0.65
40	-3.19	0	-3.08	1.15	-3.62	0	-2.77	0
50	-4.27	0.70	-3.23	1.20	-4.15	-0.22	-3.32	2.06
60	-6.13	-0.21	-5.46	1.86	-5.69	0.89	-4.33	0.87
70	-9.56	-6.98	-8.70	-4.79	-7.20	0.14	-6.57	-0.71
80	-13.96	-3.81	-13.80	-3.67	-10.29	-1.06	-9.88	-0.91
90	-11.51	-7.12	-10.91	-6.57	-5.50	-3.02	-4.95	-1.77
100	-11.60	-8.64	-11.05	-8.06	-7.65	-1.89	-6.97	-1.15
$InstPK$								
20	-0.17	0	0.13	3.36	-0.18	0.32	0.03	2.09
25	-0.02	1.83	0.09	2.42	-0.13	0.67	-0.05	1.45
30	-4.99	0	-3.54	0	-2.11	0	-0.08	2.17
35	-3.00	-0.73	-1.92	-0.32	-3.22	0.57	-2.64	0
40	-6.74	-0.67	-6.47	-0.50	-4.49	-1.49	-4.48	-0.63

In Table 10 the average deviations show that the BRKGA outperformed the DDE algorithm in all the groups of BACAP instances, both in N and $3N$ seconds. The LS only marginally reduced the deviation in some groups, so the differences in performance observed were consistently caused by the behaviour of each algorithm itself. The DDE algorithm clearly performed worse with increasing instance size in N seconds, although this effect decreased when considering $3N$ seconds. Therefore, the BRKGA seems to explore the solution space more efficiently. Nevertheless, in some instances the DDE occasionally found better solutions, albeit close to those obtained by the Genetic Algorithm in terms of cost. The results obtained for the BACASP (Table 11) lead to the same conclusions, so in summary the BRKGA seems to achieve good solutions even on large instances.

Table 11: Comparison of the BRKGA with the DDE algorithm for the BACASP on $InstMB$ and $InstPK$, considering N and $3N$ seconds as time limits

	N seconds				$3N$ seconds			
	$avgDFD$ (%)		$avgDFD_{LS}$ (%)		$avgDFD$ (%)		$avgDFD_{LS}$ (%)	
Vessels	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
$InstMB$								
20	-0.03	0	0.99	8.54	-0.10	0	0.71	10.91
30	-6.42	0	-5.11	6.97	-4.00	0	-1.63	7.66
40	-4.49	0	-3.33	3.97	-6.16	0	-3.86	5.00
50	-5.32	-3.03	-3.37	3.94	-3.77	-0.12	-2.70	0.26
60	-5.10	3.20	-2.94	4.29	-0.74	9.42	0.61	12.54
70	-14.59	-1.46	-13.06	-0.93	-9.61	-0.70	-6.14	2.57
80	-24.60	-17.97	-23.52	-16.12	-16.80	-7.79	-13.55	-4.31
90	-22.11	-17.91	-20.95	-16.41	-17.12	-12.56	-14.76	-11.53
100	-25.92	-20.64	-25.39	-20.87	-18.87	-14.42	-17.53	-13.66
$InstPK$								
20	-1.81	0	-1.63	0	-0.86	0.52	-0.40	3.72
25	-0.91	0	-0.77	0	-0.77	0	-0.65	0
30	-5.47	0	-2.25	0	-3.99	0	-1.73	0.5
35	-5.29	-1.12	-3.46	0.11	-2.99	0	-2.04	0.31
40	-5.91	8.69	-6.05	7.70	-5.38	3.30	-3.36	1.92

7 Conclusions

In this study we have addressed the Berth Allocation and quay Crane Assignment Problem (BACAP) and the Berth Allocation and specific quay Crane Assignment Problem (BACASP) in their continuous, dynamic and time-invariant variants. The objective was to develop a metaheuristic approach capable of obtaining good solutions in short times for both small and large instances. We have proposed a Biased Random-Key Genetic Algorithm with memetic characteristics, a constructive algorithm, and several Local Search procedures. Thus we developed different versions of these algorithms for both the BACAP and the BACASP, and their parameters were adjusted through extensive experiments and statistical analysis. It has been observed that the solutions based on the ordering of the vessels provided by the Genetic Algorithm are not good enough in cases in which clusters of vessels arriving at similar times and preferring similar positions on the quay have to be allocated in a globally optimal way. To address this issue, we proposed a matheuristic Local Search which can dramatically improve the solutions.

The results obtained in instances generated according to well-known criteria show that our approach is able to achieve good, and even optimal solutions, in less than 5 minutes on instances of less than 60 vessels within a planning horizon of one week, while it clearly outperforms the most efficient exact methods to date on instances with up to 100 vessels within the same time horizon. Indeed, those methods cannot even produce feasible solutions for such large instances, so our approach broadens the computational capabilities in the field for tackling both the BACAP and the BACASP.

Further future work could address other variants of the problem heuristically, such as the BACAP with variable-in-time crane assignment, which is computationally more complex and poses some problems with the number of crane movements. Another variant could address the case in which the terminal has more than a single continuous quay and the quay to which each vessel is assigned has also to be determined. Furthermore, new mathematical programming models could be developed to capture more details of the real berth allocation problem encountered at container terminals. All these attempts may enhance our tools for improving the efficiency of the terminal and the quality of service.

Acknowledgement

We kindly thank José Bermúdez and Maite León for their statistical advice. Nevertheless, responsibility for the work is entirely ours.

This study has been partially supported by the Spanish Ministry of Economy and Competitiveness, DPI2014-53665-P, cofinanced by FEDER funds, and by Generalitat Valenciana, PROMETEO/2013/049, Grant CPI-13-351.

References

Bierwirth, C. and Meisel, F. 2010. A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research* 202:615–627.

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

Blazewicz, J., Cheng, T. C. E., Machowiak, M., and Oğuz, C. 2011. Berth and quay crane allocation: a moldable task scheduling model. *Journal of the Operational Research Society* 62:1189–1197.

Chang, D., Jiang, Z., Yan, W., and He, J. 2010. Integrating berth allocation and quay crane assignments. *Transportation Research Part E: Logistics and Transportation Review* 46:975–990.

Chen, J. H., Lee, D. H., and Cao, J. X. 2012. A combinatorial benders' cuts algorithm for the quayside operation problem at container terminals. *Transportation Research Part E: Logistics and Transportation Review* 48:266–275.

Correcher, J. F., Alvarez-valdes, R., and Tamarit, J. M. 2017. A new mixed integer linear model for the berth allocation and quay crane assignment problem. Technical report, Department of Statistics and Operations Research, University of Valencia, Spain. URL: http://www.optimization-online.org/DB_HTML/2017/03/5890.html.

Elwany, M. H., Ali, I., and Abouelseoud, Y. 2013. A heuristics-based solution to the continuous berth allocation and crane assignment problem. *Alexandria Engineering Journal* 52:671–677.

Frojan, P., Correcher, J. F., Alvarez-Valdes, R., Koulouris, G., and Tamarit, J. M. 2015. The continuous Berth Allocation Problem in a container terminal with multiple quays. *Expert Systems with Applications* 42:7356–7366.

Gonçalves, J. F. and Resende, M. G. C. 2012. A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers and Operations Research* 39:179–190.

Gonçalves, J. and Resende, M. 2013. A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics* 145:500–510.

Han, X., Gong, X., and Jo, J. 2015. A new continuous berth allocation and quay crane assignment model in container terminal. *Computers and Industrial Engineering* 89:15–22.

He, J. 2016. Berth allocation and quay crane assignment in a container terminal for the trade-off between time-saving and energy-saving. *Advanced Engineering Informatics* 30:390–405.

Hopper, E. and Turton, B. 2001. An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research* 128:34–57.

Hu, Q.-M., Hu, Z.-H., and Du, Y. 2014. Berth and quay-crane allocation problem considering fuel consumption and emissions from vessels. *Computers & Industrial Engineering* 70:1–10.

Hu, Z. H. 2015. Heuristics for solving continuous berth allocation problem considering periodic balancing utilization of cranes. *Computers and Industrial Engineering* 85:216–226.

Iori, M., Martello, S., and Monaci, M. 2003. Metaheuristic algorithms for the strip packing problem. *In* P. Pardalos and V. Korotkikh (eds.), Optimization in Industry: New Frontiers. Kluwer Academic Publishers.

Iris, Ç., Pacino, D., Ropke, S., and Larsen, A. 2015. Integrated Berth Allocation and Quay Crane Assignment Problem: Set partitioning models and computational results. *Transportation Research Part E: Logistics and Transportation Review* 81:75–97.

Karam, A. and Eltawil, A. B. 2016. Functional integration approach for the berth allocation, quay crane assignment and specific quay crane assignment problems. *Computers and Industrial Engineering* 102:458–466.

Lalla-Ruiz, E., González-Velarde, J. L., Melián-Batista, B., and Moreno-Vega, J. M. 2014. Biased random key genetic algorithm for the Tactical Berth Allocation Problem. *Applied Soft Computing Journal* 22:60–76.

Le, M., Wu, C., and Zhang, H. 2012. An integrated optimization method to solve the berth-QC allocation problem. *In* 2012 8th International Conference on Natural Computation, pp. 753–757. IEEE.

Le, M. N., Ong, Y. S., Jin, Y., and Sendhoff, B. 2009. Lamarckian memetic algorithms: Local optimum and connectivity structure analysis. *Memetic Computing* 1:175–190.

Li, W., Wu, Y., and Goh, M. 2015. Planning and Scheduling for Maritime Container Yards, pp. 1–110. *In* Planning and Scheduling for Maritime Container Yards: Supporting and Facilitating the Global Supply Network, chapter 2. Springer International Publishing, Cham.

Lim, A. 1998. The berth planning problem. *Operations Research Letters* 22:105–110.

Maxwell, S. E. and Delaney, H. D. 2004. Designing Experiments and Analyzing Data: a Model Comparison Perspective. Lawrence Erlbaum Associates, Mahwah, New Yersey, second edition.

Meisel, F. and Bierwirth, C. 2009. Heuristics for the integration of crane productivity in the berth allocation problem. *Transportation Research Part E: Logistics and Transportation Review* 45:196–209.

Meisel, F. and Bierwirth, C. 2013. A Framework for Integrated Berth Allocation and Crane Operations Planning in Seaport Container Terminals. *Transportation Science* 47:131–147.

Nishimura, E., Imai, A., and Papadimitriou, S. 2001. Berth allocation planning in the public berth system by genetic algorithms. *European Journal of Operational Research* 131:282–292.

Pan, Q.-K., Tasgetiren, M., and Liang, Y.-C. 2007. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *In* Proceedings of the genetic and evolutionary computation conference 2007 (GECCO07), pp. 126–133.

Pan, Q.-K., Tasgetiren, M. F., and Liang, Y.-C. 2008. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers and Industrial Engineering* 55:795–816.

Park, Y. and Kim, K. 2003. A scheduling method for berth and quay cranes. *OR Spectrum* pp. 1–23.

Raa, B., Dullaert, W., and Schaeren, R. V. 2011. An enriched model for the integrated berth allocation and quay crane assignment problem. *Expert Systems with Applications* 38:14136–14147.

Rashidi, H. and Tsang, E. P. K. 2013. Novel constraints satisfaction models for optimization problems in container terminals. *Applied Mathematical Modelling* 37:3601–3634.

Rodriguez-Molins, M., Salido, M. A., and Barber, F. 2014a. A GRASP-based metaheuristic for the Berth Allocation Problem and the Quay Crane Assignment Problem by managing vessel cargo holds. *Applied Intelligence* 40:273–290.

Rodriguez-Molins, M., Salido, M. A., and Barber, F. 2014b. Robust Scheduling for Berth Allocation and Quay Crane Assignment Problem. *Mathematical Problems in Engineering* 2014.

Shang, X. T., Cao, J. X., and Ren, J. 2016. A robust optimization approach to the integrated berth allocation and quay crane assignment problem. *Transportation Research Part E: Logistics and Transportation Review* 94:44–65.

Stahlbock, R. and Voss, S. 2008. Operations research at container terminals: A literature update. *OR Spectrum* 30:1–52.

Storn, R. and Price, K. 1997. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11:341–359.

Tasgetiren, M., Pan, Q.-K., Liang, Y.-C., and Suganthan, P. 2007a. A discrete differential evolution algorithm for the total earliness and tardiness penalties with a common due date on a single machine. *In* Proceedings of the 2007 IEEE symposium on computational intelligence in scheduling (CISched2007), pp. 271–278.

Tasgetiren, M., Pan, Q.-K., Suganthan, P., and Liang, Y.-C. 2007b. A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flowtime criterion. *In* Proceedings of the 2007 IEEE symposium on computational intelligence in scheduling (CISched2007), pp. 251–258.

Türkoğullari, Y. B., Taşkin, Z. C., Aras, N., and Altinel, I. K. 2014. Optimal berth allocation and time-invariant quay crane assignment in container terminals. *European Journal of Operational Research* 235:88–101.

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

UNCTAD 2016. Review of Maritime Transport. United Nations Conference on Trade and Development.

Xiao, L. and Hu, Z. H. 2014. Berth allocation problem with quay crane assignment for container terminals based on rolling-horizon strategy. *Mathematical Problems in Engineering* 2014.

Yang, C., Wang, X., and Li, Z. 2012. An optimization approach for coupling problem of berth allocation and quay crane assignment in container terminal. *Computers & Industrial Engineering* 63:243–253.

Zhang, C., Zheng, L., Zhang, Z., Shi, L., and Armstrong, A. J. 2010. The allocation of berths and quay cranes by using a sub-gradient optimization technique. *Computers & Industrial Engineering* 58:40–50.

Appendix A An MILP for the BACAP

This model was presented in Correcher et al. (2017). Here we propose a modified version in which the cost of exceeding the planning horizon is taken into consideration.

The following variables are defined:

t_i = berthing time of vessel i

p_i = berthing position of vessel i

$$r_{iqt} = \begin{cases} 1, & \text{if the handling of vessel } i \text{ with } q \text{ cranes starts at time } t \\ 0, & \text{otherwise} \end{cases}$$

$$\sigma_{ij} = \begin{cases} 1, & \text{if vessel } i \text{ is completely to the left of vessel } j\text{, that is,} \\ & \text{vessel } i \text{ is completely processed before vessel } j \\ 0, & \text{otherwise} \end{cases}$$

$$\delta_{ij} = \begin{cases} 1, & \text{if vessel } i \text{ is completely below vessel } j\text{, that is, vessel } i \text{ is} \\ & \text{completely to the right of vessel } j\text{, looking at them from the quay} \\ 0, & \text{otherwise} \end{cases}$$

h_i = delay incurred in the handling of vessel i

e_i = deviation of vessel i from its desired position

exc_i = difference between the berthing time of vessel i and the planning horizon

These variables and the parameters defined in Section 3 are the elements of the following model:

$$Min \sum_{i \in V} (C_i^w (t_i - a_i) + C_i^d h_i + C_i^p e_i + C_i^h exc_i) \quad (2)$$

s. t.:

$$\sum_{t=a_i}^{H} \sum_{q=q_i^{min}}^{q_i^{max}} r_{iqt} = 1, \qquad \forall i \in V \quad (3)$$

$$t_i = \sum_{t=a_i}^{H} \sum_{q=q_i^{min}}^{q_i^{max}} r_{iqt} \cdot t, \qquad \forall i \in V \quad (4)$$

$$t_j - t_i - \sum_{t=a_i}^{H} \sum_{q=q_i^{min}}^{q_i^{max}} (r_{iqt} \cdot u_i^q) - (\sigma_{ij} - 1)H \geq 0, \qquad \forall i, j \in V, i \neq j \quad (5)$$

$$p_j - (p_i + l_i) - (\delta_{ij} - 1)L \geq 0, \qquad \forall i, j \in V, i \neq j \quad (6)$$

$$\sigma_{ij} + \sigma_{ji} + \delta_{ij} + \delta_{ji} \geq 1, \qquad \forall i, j \in V, i \neq j \quad (7)$$

$$p_i + l_i \leq L + 1, \qquad \forall i \in V \quad (8)$$

$$\sum_{i \in V} \sum_{q=q_i^{min}}^{q_i^{max}} \sum_{\tau=max(a_i, t-u_i^q+1)}^{t} r_{iq\tau} \cdot q \leq Q, \qquad \forall t \in T \quad (9)$$

$$h_i \geq t_i - s_i + \sum_{t=a_i}^{H} \sum_{q=q_i^{min}}^{q_i^{max}} (r_{iqt} \cdot u_i^q) - 1, \qquad \forall i \in V \quad (10)$$

$$e_i \geq p_i - d_i, \qquad \forall i \in V \quad (11)$$

$$e_i \geq d_i - p_i, \qquad \forall i \in V \quad (12)$$

$$\sigma_{ij}, \delta_{ij}, \in \{0, 1\}, \qquad \forall i, j \in V, i \neq j \quad (13)$$

$$r_{iqt} \in \{0, 1\}, \qquad \forall i \in V, \forall t \in \{a_i, \ldots, H\}, \forall q \in \{q_i^{min}, \ldots, q_i^{max}\} \quad (14)$$

$$h_i, e_i \geq 0, \qquad \forall i \in V \quad (15)$$

$$p_i \geq 1, \qquad \forall i \in V \quad (16)$$

$$exc_i \geq t_i - H, \qquad \forall i \in V \quad (17)$$

$$exc_i \geq 0, \qquad \forall i \in V \quad (18)$$

Appendix B A cutting plane algorithm for BACASP

The following cutting plane algorithm was proposed in Correcher et al. (2017):

Algorithm 1 Cutting plane algorithm for the BACAP–BACASP

```
Require: A formulation $P$ for BACAP. See Appendix A
Ensure: A solution for the BACAP without improper complete sequences
 1: Solve $P$
 2: while there is at least one improper complete sequence in an optimal solution of $P$ do
 3:   for all improper complete vessel sequence $S$ found do
 4:     Insert in $P$ a cut for $S$ based on (19)
 5:     if $S \leq MaxSizeSeqForPerm$ vessels then
 6:       Insert in $P$ such a cut for each other permutation of the vessels in $S$
 7:     end if
 8:   end for
 9:   Solve $P$
10: end while
```

The following constraint prevents the vessels in the sequence $S = s_1, \ldots, s_n$ from being assigned forming an improper complete sequence in that specific order in any time and position:

$$\sum_{i \in S} \sum_{t=a_i}^{H} \sum_{q=q_i^{min}}^{q_i^{max}} q\ r_{iqt} \leq Q + M \left(\sum_{j=s_1}^{s_{n-1}} (\sigma_{j,j+1} + \sigma_{j+1,j} - \delta_{j,j+1}) + n - 1 \right) \tag{19}$$

considering that:

$$M = \sum_{i \in S} q_i^{max} - Q$$