Applied Soft Computing 22 (2014) 60–76

Contents lists available at ScienceDirect

Applied Soft Computing

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

Biased random key genetic algorithm for the Tactical Berth Allocation Problem

Eduardo Lalla-Ruiz [a], José Luis González-Velarde [b], Belén Melián-Batista [a], J. Marcos Moreno-Vega [a,*]

[a] Universidad de La Laguna, Dpto. de Estadística, IO y Computación, 38271 La Laguna, Spain
[b] Instituto Tecnológico de Monterrey, Mexico

ARTICLE INFO

Article history:
Received 28 July 2013
Received in revised form 19 March 2014
Accepted 23 April 2014
Available online 10 May 2014

Keywords:
Container terminal
Berth allocation
Biased random keys
Genetic algorithm

ABSTRACT

The Tactical Berth Allocation Problem (TBAP) aims to allocate incoming ships to berthing positions and assign quay crane profiles to them (i.e. number of quay cranes per time step). The goals of the TBAP are both the minimization of the housekeeping costs derived from the transshipment container flows between ships, and the maximization of the total value of the quay crane profiles assigned to the ships. In order to obtain good quality solutions with considerably short computational effort, this paper proposes a biased random key genetic algorithm for solving this problem. The computational experiments and the comparison with other solutions approaches presented in the related literature for tackling the TBAP show that the proposed algorithm is applicable to efficiently solve this difficult and essential container terminal problem. The problem instances used in this paper are composed of both, those reported in the literature and a new benchmark suite proposed in this work for taking into consideration other realistic scenarios.

Introduction

The competition between sea-freight container terminals has hugely grown due to the increased transport of goods along the main maritime routes of containers and the important economic role played by the container ports in countries and regions. According to a United Nations Conference on Trade And Development (UNCTAD) study [30], the average container traffic growth in the last two decades is around 10%. This growth of containers traffic leads to port terminals to manage a larger number of incoming ships at a competitive price. In order to achieve this goal, container terminals are forced to improve their management capabilities and resources utilization with the objective of enhancing productivity to compete with other port terminals. For this reason, an inefficient utilization of some key-factor resources like berths and quay cranes (QCs) might be translated into a delay of the yard-side and land-side operations, resulting in lower global productivity of the container terminal.

From a general point of view, the loading and unloading processes in a container terminal consist of several phases as indicated by Steenken et al. [36], Stahlbock et al. [35] and Monaco et al. [25]. Whenever a ship arrives to the port, it is allocated to a specific berth, where a set of quay cranes are required to perform its loading and unloading operations. The unloaded containers are stored on the yard in order to continue their route by trains, trucks or other ships. Meanwhile, the allocated ship may be loaded with transshipment containers which will continue their route to other container terminals. The processes described above present a high dependence among the tasks and operations that are carried out in them. Concerning the operations of the Tactical Berth Allocation Problem studied in this paper, the ships are allocated in berthing positions and the time of their stay in port depends on the number of quay cranes assigned for serving them. In this regard, the quay crane productivity depends on the number of ships assigned to it, while the berths productivity depends on the waiting time and service time of the ships. According to Expósito et al. [12], most of the time spent by a container ship in a terminal is used to perform the loading and unloading tasks by quay cranes. Given this important relationship between these operations, and as indicated by Bierwirth et al. [4], integrating both logistic problems is a common task in the management of container terminals. Their effective resolution will suppose important savings for the container terminal due to the fact that the

* Corresponding author. Tel.: +34 922318175.
E-mail addresses: elalla@ull.es (E. Lalla-Ruiz), gonzalez.velarde@itesm.mx (J.L. González-Velarde), mbmelian@ull.es (B. Melián-Batista), jmmoreno@ull.es (J.M. Moreno-Vega).

http://dx.doi.org/10.1016/j.asoc.2014.04.035
1568-4946/

berthing operations are one of the tasks with the highest impact on the service costs.

Regarding modern container terminals, one of the most important problems is the coordination between the storage of transshipment containers and the loading and unloading of ships. When terminal managers attempt to address this problem they also pursue at the same time to reduce the unnecessary use of transport resources. Furthermore, they also consider the direct dependence of the processes involved, specially with the allocation of cranes as they are responsible for the loading/unloading of containers. In this concern, as the number of cranes assigned to ship docked at a determined berth is reduced, the stay of the ship at port lengthens. This can cause a deviation of the berthing position or scheduling of an incoming ship that needs to pick up some transshipment containers deposited at that berth. The consideration of this issue and its appropriately resolution would suppose an improvement in container traffic management, thus the transshipment traffic comprises an increasing proportion of the maritime terminal total traffic. Moreover, this improvement in the management results in an advantage that is frequently taken into account in the decision making of the shipping lines and shippers since the terminal transshipment capacity is a decisive characteristic as indicated by Bevskovnik et al. [3].

This paper proposes a new alternative to solve the TBAP involved processes (berth and quay allocation) jointly. The proposed method is based on the development of a biased random key genetic algorithm (BRKGA). Its goal is to provide a high-quality solution in short computational times with the aim of providing terminal managers a useful tool which will make berth and quay crane allocation decisions more suitable and efficient. The effectiveness of the approach has been assessed by comparing its results with the best-known algorithmic methods reported in the related literature. Furthermore, in order to test the efficiency of the BRKGA algorithm in different scenarios, a set of new instances is proposed.

The rest of the paper is organized as follows. A literature review is provided in *Literature review*. The TBAP description is presented in *The Tactical Berth Allocation Problem*. The algorithm proposed in this paper is introduced in *Biased random key genetic algorithm*. *Computational results* section is devoted to summarize the computational experiments carried out in this work. Final section describes the conclusion.

Literature review

Several studies about maritime container terminals focus their attention on the strategic, tactical and operational problems in a container port. Some of these studies are those presented by Murty et al. [27], Vis et al. [39], Stahlbock et al. [35], Steeken et al.[36], Günter et al. [15] and Crainic et al. [8]. Some problems related to the TBAP can be highlighted: Berth Allocation Problem (BAP), that consists of assigning ships to berths over the time horizon, and Quay Crane Allocation Problem (QCAP), that consists of assigning quay cranes to ships with the purpose of serving them. A summarized state of the art of these problems and their integrations is presented by Bierwirth et al. [4], where a classification according to the way in which the integration of these two problems is performed is introduced.

The Berth Allocation Problem has been extensively studied in the literature. Due to the diversity of maritime terminal layouts, research has produced multitude of considerations for the BAP. Concerning the arrival time of the ships, two types of problems can be considered, the static version (SBAP) presented by Imai et al. [19] and the dynamic version (DBAP) presented by Imai et al. [20]. In the first case, ships are in port before the planning horizon begins, whereas in the second case, ships can arrive at any moment of the planning horizon. The SBAP can be solved in polynomial time since it can be reduced to a classical assignment problem which is known to be polynomially-solvable (Pinedo [32]). The SBAP is extended to the dynamic version by Imai et al. [20]. Due to the difficulty of accurately finding an optimal solution, these authors develop and present a heuristic using the sub-gradient method with a Lagrangian relaxation. Monaco and Sammarra [25] present a stronger formulation for the model proposed by Imai et al. [20] and present a Langrangian relaxation with a non-standard multiplier adjustment method for solving it. Nishimura et al. [28] extend this problem with the consideration of different water depths. Lalla et al. [22] present an effective and efficient Tabu Search strategy with Path-Relinking for solving this problem. Moreover, Imai et al. [21] include ships priorities.

Concerning spatial constraints, the Berth Allocation Problem can be classified as discrete, continuous or hybrid (see Bierwirth et al. [4]). The discrete case presents a quay divided into sections referred to as berths (see Imai et al. [20], Cordeau et al. [7], Nishimura et al. [28], Hansen et al. [16]). In the continuous case, there is no division of the quay, so that an incoming ship can be assigned to the quay taking into account its spatial measures (see Lim [24], Park et al. [31], Wang et al. [40]). In the hybrid version of the BAP, the quay is divided into berths, but a ship can occupy more than one berth (some examples can be found in the work presented by Cordeau et al. [7], Imai et al. [18], Cheong et al. [5]).

Cordeau et al. [7] introduce two formulations for the DBAP, in which both discrete and hybrid quay are considered. In order to solve them, two Tabu Search heuristics are presented. Moreover, Hansen et al. [16] take into account the costs for waiting and handling as well as earliness or tardiness of completion with the purpose of including priorities. There is also a handling cost associated to each berth that would vary depending on which berth is used. A Variable Neighbourhood Search algorithm is then developed for its resolution, which attains the optimal solution in the vast majority of cases.

The Quay Crane Allocation Problem (QCAP) constitutes a problem that has hardly received attention from researchers. However, it becomes important when it is integrated into the BAP, since the service time of the ships depends on the number of quay cranes assigned to them. Several studies that tackle the integration of both Berth Allocation and Quay Crane Allocation Problems can be found in the literature. The most influential papers related to the Tactical Berth Allocation Problem are described below. Imai et al. [18] propose a model for simultaneous berth-crane allocation that minimizes the total service time and develop a genetic algorithm for its solution. In their study, they do not consider the relationship between the handling time and the number of cranes. Therefore, a ship will start to be served only when a predetermined number of cranes becomes available. Otherwise, the ship will have to wait until they are available. Zhang et al. [41] consider the allocation of discrete berths and quay cranes for ships arriving at container terminals. For this purpose, a mixed integer programming model is presented and solved by a sub-gradient optimization algorithm. In this integrated model, the berth and quay crane allocation problems are simultaneously solved under the consideration of coverage ranges of the quay cranes and limited adjustments of the quay cranes allocated during loading and unloading. Liang et al. [23] address the operational problem of determining the berthing position and the number of quay cranes assigned to the ships. They aim to minimize the sum of the handling, waiting and delay times for every ship. Due to its difficulty they propose a genetic algorithm to find an approximate solution. Raa et al. [33] present a model for the integration of the BAP and QCAP at an operational level where ships priorities, preferred berthing locations and handling times are taken into account.

While the majority of the literature works related to the integration of the BAP and QCAP are focused on an operational level, only

a few contributions on tactical planning are available. However, many of these tactical problems are solved intuitively by terminal planners. Due to this, the application of solution approaches becomes important since they can offer improvements in terms of efficiency and productivity of the container terminal. In order to tackle this issue, the Tactical Berth Allocation Problem is introduced by Giallombardo et al. [13] as an integration of the BAP and QCAP at a tactical level. In that work, some operational constraints in the definition of the tactical problem, such as rules, common policies and best practices ensuring robustness and realism to the tactical planning, are taken into consideration. The solution approach for this integration is performed sequentially; i.e. in the first place, quay cranes are assigned to ships, then the ships are scheduled and assigned to the berths, then this process restarts and it is repeated a certain number of iterations. Moreover, for solving this problem, a Branch&Price and a Tabu Search are developed. The computational results show that for larger instances these heuristics need much computational time to give a feasible quality solution. Furthermore, for two instances the heuristic is not able to provide a feasible solution. Another work related to the TBAP is the one presented by Vacca et al. [38], which proposes an exact branch-and-price algorithm for solving the TBAP that employs several accelerating techniques for the master and pricing problems. They make use of two different initializations of the master problem; one based on artificial variables that satisfy some constraints related to the model included in that work ($B\&P$) and another provided by the algorithm developed by Giallombardo et al. [13] ($B\&P$+INIT), where the solution given by this method is added to the master problem as a root node. The computational experiments were performed on instances derived from the test set introduced by Giallombardo et al. [13] with some changes on the problem data in order to avoid symmetries, which slow down the convergence of the exact algorithms.

Other works related to the TBAP can be found in the literature (see Cordeau et al. [6], Moorthy et al. [26]). They can be taken as an introduction to the TBAP. Moorthy et al. [26] introduce a framework for the tactical home berth allocation problem, which concerns the allocation of favourite berthing locations (home berths) to ships which periodically call at the terminal. The design of the framework for the home berth allocation problem is modelled as a bi-criteria optimization problem, taking into consideration the trade-off between operational costs and service levels. In order to evaluate the usefulness of this approach, they simulate the performance of the template using a dynamic berth allocation package developed in [9]. Cordeau et al. [6] present the Service Assignment Problem (SAP), where a service is the sequence of ports visited by a ship. The objective is to minimize the container rehandling operations inside the yard assigning the berth. The SAP is quite similar to the TBAP. Besides the fact that both problems are considered at a tactical level. The SAP can be seen as a relaxation of the TBAP if the use of quay crane profiles is discarded and the time dimension is collapsed.

The Tactical Berth Allocation Problem

The Tactical Berth Allocation Problem (TBAP) proposed by Giallombardo et al. [13] pursues to determine the berthing position, berthing time and allocation of quay cranes (through the use of quay crane profiles) for container ships arriving to the port over a well-defined time horizon.

Since the number of quay cranes assigned to serve a ship determines the duration that a ship stays at a particular berth, it makes sense to consider the service time for each ship as a function of the number of quay cranes assigned to it. In order to model this feature, the quay crane profiles are introduced. These are a representation of the number of quay cranes (QC) that will be assigned to a certain ship from its arrival to its departure. Thus, in the TBAP we are given a set of QC profiles P_i for each ship $i \in N$ that consider the number of containers to be handled in the ship and the amount of QC hours requested by the ship owner. Each quay crane profile has a specified value, which reflects technical aspects (such as the resources utilized), but that is also computed by taking into account the specific ship that will use the profile. In other words, the same QC profile can have different values when applied to different ships, according to their priority or importance. The interested reader is referred to Giallombardo et al. [13] for a detailed description of quay crane profiles.

Table 1
Quay crane profiles.

Ship 1							
Profile 1	1	2	2	3	2	1	$v_1^1 = 41{,}965$
Profile 2	2	3	4	2			$v_1^2 = 42{,}035$
Profile 3	3	4	4				$v_1^3 = 42{,}236$
Ship 2							
Profile 1	1	1	2	1	1		$v_2^1 = 30{,}540$
Profile 2	1	2	3				$v_2^2 = 31{,}245$
Profile 3	1	1	2	2			$v_2^3 = 31{,}005$
Ship 3							
Profile 1	3	3	4				$v_3^1 = 40{,}780$
Profile 2	2	2	1	2	3		$v_3^2 = 39{,}890$
Profile 3	2	3	3	2			$v_3^3 = 40{,}120$
Ship 4							
Profile 1	1	2	2	3	4		$v_4^1 = 43{,}120$
Profile 2	4	4	4				$v_4^2 = 44{,}315$
Ship 5							
Profile 1	4	4	3	1			$v_5^1 = 45{,}230$
Profile 2	1	1	1	2	3	4	$v_5^2 = 42{,}390$
Profile 3	2	2	2	3	3		$v_5^3 = 43{,}234$
Ship 6							
Profile 1	1	2	3				$v_6^1 = 31{,}900$
Profile 2	3	3					$v_6^2 = 32{,}675$
Profile 3	3	2	1				$v_6^3 = 31{,}900$
Profile 4	1	2	2	1			$v_6^4 = 31{,}600$

Table 1 shows an example of the QC profiles structure for a set of ships. The distribution of QCs that will serve the ships along their stay and the QC profile value are depicted. In this table, the total service of the profile 1 for ship 1 will be equal to six time steps, so that, the ship will be served by one, two, two, three, two and one QCs, respectively. Moreover, the total value of the profile will be equal to 41,965. When the distribution of the assigned QCs increases, for example the profile 2, the total service time required by the ship is reduced and the profile value is incremented to 42,035.

When a ship arrives at a berth, some of the containers it carries might be in the middle of a route waiting to continue their trajectory into another ship. Thus, terminal managers have to allocate outgoing containers into positions close to the receiving ships. The operations involved in the transfer of containers from yard-to-yard positions are referred to as housekeeping. The housekeeping process appears when there are transshipment containers that are unloaded in a certain position in the quay and need to be moved to positions near to the incoming ship in which they will continue their route. This process requires an efficient management in order to avoid wasting transport resources unnecessarily and delays in the ship service times. For this purpose, the housekeeping yard cost is included in the TBAP as a part of the cost function that depends on the distance among the incoming ships that will exchange transshipment containers.

An example of the housekeeping costs for three berths are shown in Table 2, where the movement of transshipment containers between two ships berthed in the same quay is less than the

Table 2
Housekeeping costs.

Berth	1	2	3
1	11	14	16
2	14	12	16
3	18	16	13

transshipment of container between different berthing positions among them.

As mentioned above, the model formulation of the Tactical Berth Allocation Problem used in this work corresponds to the one proposed by Giallombardo et al. [13]. The following parameters are used to define the problem:

N set of ships
M set of berths
H set of time steps
P_i set of feasible quay crane profiles for the ship $i \in N$
t_i^p service time of ship $i \in N$ under QC profile $p \in P_i$
v_i^p the value of serving the ship $i \in N$ with the QC profile $p \in P_i$
Q maximum number of quay cranes available
f_{ij} flow of containers exchanged between ships $i, j \in N$
d_{kw} housekeeping cost per unit of container between yard slots in berths $k, w \in M$
a_i, b_i earliest, latest arrival time of ship $i \in N$
a^k, b^k start, end of the availability time of the berth $k \in M$

The decision variables used are shown below:

$y_i^k \in \{0, 1\}, \forall k \in M, \forall i \in N$, set to 1 if ship i is assigned to berth k, and 0 otherwise.
$\lambda_i^p \in \{0, 1\}, \forall p \in P_i, \forall i \in N$, set to 1 if ship i is served under profile p, and 0 otherwise.

The assumptions contemplated in the mathematical model are the following:

(a) Each berth $k \in M$ can only handle one ship at a time.
(b) The service time of each ship $i \in N$ is determined by the quay crane profile $p \in P_i$ assigned to it.
(c) One and only one QC profile can be assigned to each ship.
(d) Each ship $i \in N$ can be serviced only after its arrival time a_i.
(e) Each ship $i \in N$ have to be serviced until its departure time b_i.
(f) Each ship $i \in N$ can only be docked at berth $k \in M$ after it becomes available at time step a^k.
(g) Each ship $i \in N$ can only be docked at berth $k \in M$ until it becomes unavailable at time step b^k.
(h) At every time step the total number of assigned quay cranes cannot exceed the maximum number of quay cranes Q available at the terminal.

The objective function (1) aims to maximize the sum of the values of the chosen quay crane profiles v_i^p assigned to all the ships $i \in N$ and, simultaneously, minimize the yard-related housekeeping cost generated by the flows of container exchanged among the ships. Note that the objective function value is calculated from the terminal viewpoint. It is aimed, on the one hand, to increase the benefit of the terminal through the quay crane profiles assignment. On the other hand, to reduce the housekeeping costs that the terminal has to afford related to the movement of transshipment containers from the berth to another

$$\max \sum_{i \in N} \sum_{p \in P_i} \lambda_i^p v_i^p - \frac{1}{2} \sum_{i \in N} \sum_{k \in M} y_i^k \sum_{j \in N} \sum_{w \in M} f_{ij} d_{kw} y_j^w \quad (1)$$

In the example shown in Fig. 1 a solution example for the TBAP is illustrated. It consists of $|N| = 6$ ships, $|M| = 3$ berths and a maximum number of available quay cranes $Q = 6$ over a time horizon of 14 time steps. Table 3 reports the information regarding the time windows

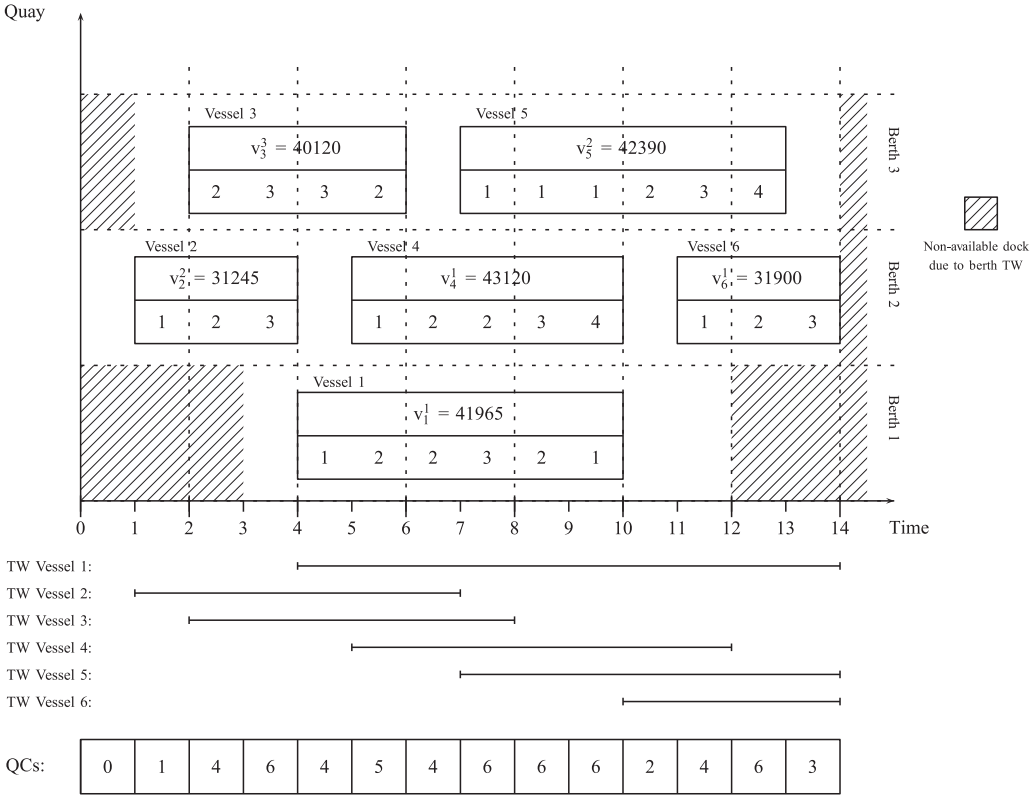

Fig. 1. Solution example for the TBAP.

Table 3
Berths and ships time windows.

Ship	1	2	3	4	5	6
a_i	4	1	2	5	7	10
b_i	14	7	8	12	14	14

Berth	1	2	3
a^i	3	0	1
b^i	12	14	14

Table 4
Transshipment of containers.

Ship	1	2	3	4	5	6
1	0	0	0	100	0	100
2	0	0	50	150	75	125
3	0	50	0	50	200	0
4	100	150	50	0	0	75
5	0	75	200	0	0	40
6	100	125	0	75	40	0

of the ships and berths. Table 2 depicts the information related to the housekeeping costs per container. Table 4 reports the information related to the interchange of containers among the ships. The information concerning the possible profiles for each container ship is depicted in Table 1. The objective value for this solution example is 217,700, where 13,040 corresponds to the housekeeping cost and 230,740 to the total profile value. Furthermore, consider for instance ship 6, although it is able to dock at time step 10, if so planned, this would incur in a quay crane infeasibility since it would be necessary 7 QCs at time step 12 to perform this schedule.

Biased random key genetic algorithm

Genetic algorithms (GAs) are bio-inspired algorithms that use the concepts of biological evolution and survival of the fittest for obtaining optimal or near optimal solutions in optimization problems (Holland [17]). As an analogy, a solution is presented as an individual within a set of solutions known as *population*. The information defining each individual is contained in a *chromosome*, which is composed of *genes* that can take different values, *alelles*, with the aim of identifying the solution. The chromosomes have an associated fitness value, which depend on some objective function. The GAs evolve the population along iterations known as *generations*. Each new population is built from the previous population using different operators, the most common are *selection*, *mutation* and *crossover*. The new population is generally designed to preserve the genetic material of the better solutions (i.e. survival of the fittest). This concept establishes a preference for the chromosomes with better fitness.

The concept of random keys is introduced by Bean [2] with the aim of avoiding the typical difficulty of general genetic algorithms to maintain feasibility of solutions from parents to offspring. He proposes to overcome this difficulty by representing the solutions through random keys. A random key is a real-valued number in the interval [0, 1). Thus, the chromosomes are encoded as vectors of n alleles over the interval [0, 1) (n depends on the optimization problem considered). In order to associate each chromosome with a valid solution of the optimization problem, it is used a deterministic procedure called *decoder*. It is a procedure that transform a random key vector in a feasible solution of the problem. A random key genetic algorithm (RKGA) is a genetic algorithm that codifies the solutions as random keys.

The biased random key genetic algorithm (BRKGA) is presented by Ericsson et al. [11] as a variation of the RKGA. BRKGA differs from RKGA in the way the crossover is performed. In a BRKGA, the population is divided in two subpopulations, the elite subpopulation and non-elite subpopulation. The elite subpopulation contains a small set of the best solutions of the population. The rest of solution belongs to the non-elite subpopulation. To generate the offspring, BRKGA selects one parent from the elite subpopulation and the other parent from the rest of the population. Moreover, in the crossover process, for giving more probability to the elite parent genes, biased coin favouring the elite parent is tossed, so the child would have more probability of inhering the keys of its elite parent. Although this specialized version of the algorithm did not receive a special name since it was proposed as a heuristic to solve a particular problem, it contained the germ of what, in a subsequent paper by Gonçalves and Resende [14] would be identified as a general purpose metaheuristic: Biased Random Key Genetic Algorithm.

Specifically, in a BRKGA the solutions are encoded as vectors of n random keys, and evolve a population, Pop, of $|Pop|$ vectors of random keys. The fitness value of each chromosome is obtained by applying a decoder. At each iteration k, the random key vectors are decoded and the population is partitioned into two sets: Pop_e consisting of $|Pop_e|$ vectors made up of the best solutions by means of the objective function value and Pop_c consisting of $|Pop| - |Pop_e|$ vectors made up of $Pop_c = Pop - Pop_e$. Population $k+1$ contains all $|Pop_e|$ elite-set solutions of population k as well as a set Pop_m consisting of $|Pop_m|$ newly-generated mutants, with $|Pop| - |Pop_e| - |Pop_m| \geq 0$. A mutant is a vector of n random keys used to help the algorithm avoid getting trapped in local optima. The remaining $|Pop| - |Pop_e| - |Pop_m|$ are produced by combining pairs of solutions (vectors rka and rkb) from population k using uniform parameterized crossover of Spears and DeJong [34], where the i-th key, $rko(i)$, of the offspring inherits the i-th key of parent rka, that is, $rko(i) = rka(i)$ with probability $prob_h$, and the i-th of parent rkb with probability $1 - prob_h$. Furthermore, in a BRKGA, $prob_h > 0.5$, so that the child has a greater probability of inheriting the key of its elite parent than that of its non-elite parent.

Algorithm 1. Biased random key genetic algorithm

```
BRKGA (|Pop|, |Pop_e|, |Pop_m|, n, prob_h)
Generate population Pop with individuals having n random-keys;
while (stopping criterion not satisfied) do
    Evaluate fitness of each new individual in Pop
    Partition Pop into sets Pop_e and Pop_c
    Generate mutants Pop_m each having n random keys;
    Pop+ = Pop_e ⋃ Pop_m
    for (i = 1 to |Pop| − |Pop_e| − |Pop_m|) do
        Select a parent rka at random from Pop_e
        Select a parent rkb at random from Pop_c
        for (j = 1 to n) do
            Toss biased coin having probability prob_h > 0.5 of heads
            if (Toss = heads) then
                rko[j] = rka[j]
            else
                rko[j] = rkb[j]
        end
        Pop+ = Pop+ ⋃ {rko}
    end
    Pop = Pop+
end
return pop* = argmin{f(pop)|pop ∈ Pop}
```

The BRKGA is summarized in the pseudo-code of Algorithm 1. It takes as input the size of the population, $|Pop|$, elite set, $|Pop_e|$, mutant set, $|Pop_m|$ (where $|Pop_e| + |Pop_m| \leq |Pop|$ and $2 \times |Pop_e| \leq |Pop|$), the size of the random-key vector (n), and the

probability that the toss results in heads ($prob_h > 0.5$). In line 2, the initial population is generated, consisting of $|Pop|$ vectors, each with n real-valued keys randomly generated in the interval (0, 1]. The iterations of the algorithm correspond to the loop in lines 3 to 21. In line 4, the fitnesses of all newly added individuals of population *Pop* are evaluated. Population *Pop* is partitioned into a smaller set Pop_e of elite individuals (those with the best overall fitness values) and a larger set Pop_c with the remaining population, that is, $Pop_c = Pop \setminus Pop_e$. In line 6 the mutant set Pop_m is generated in the same way that the initial population was generated. It is added to the population of the next generation together with the elite set in line 7. The remainder of the population of the next generation is completed in lines 8 to 19. Parents *rka* and *rkb* are selected at random in lines 9 and 10, respectively. Then, the parameterized uniform crossover is applied in lines 11 to 17 to produce the offspring *rko*, which is added to the population of the next generation in line 18. An iteration is completed in line 20 by making the population of the current generation that of the next generation. Finally, after the stopping criterion has been satisfied, the fittest individual of population *Pop* is returned by the algorithm in line 22.

BRKGA for the Tactical Berth Allocation Problem

This section is devoted to describe the configuration of the initial population, the way in which the solutions are encoded, the decoding process and the stopping criterion of the BRKGA for the TBAP.

Initial population

The initial population is composed by $|Pop|$ chromosomes, each randomly generated and satisfying the feasibility criteria imposed by the problem.

Encoding

The chromosomes are represented as vectors of randomly generated real numbers in [0, 1), whose length is twice the number of ships $|N|$. The first $|N|$ positions represent the ships service order. The last $|N|$ positions correspond to the assignment of profiles to ships. Fig. 2(a) shows an example of the structure of a chromosome for 6 ships, l whose quay crane profiles are those reported in Table 1.

Decoding

In order to decode a random key based solution for the TBAP, the chromosome is divided into two subsequences. The first subsequence of $|N|$ genes is used to determine the preference berthing order vector v, which stores an ordered list of the priority of the ships to be used when performing the assignation of berths. For this purpose, the alleles are sorted in an increasing order and then

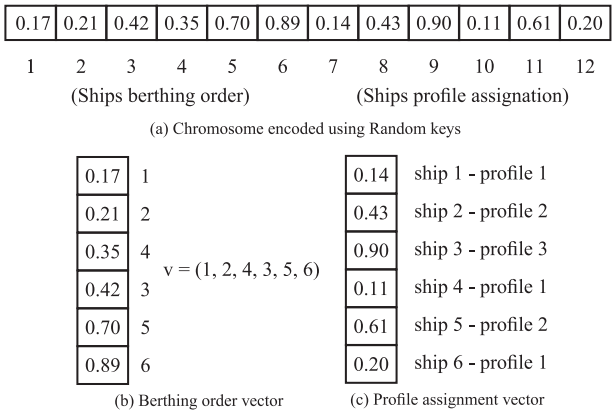

Fig. 2. Random keys decodification process example.

a berthing order is determined following the sorted subsequence. The ship identification depends on the position of each allele in the unsorted subsequence. Thus, the first ship to be allocated is the one with the lowest allele and the position that this allele occupies in the disordered sequence determines the number of the ship. Fig. 2(b) shows the first increasingly ordered sequence of $|N|$ items. In this case, the allele with a value of 0.17 identifies the first ship to be berthed, which in this case corresponds to ship 1 due to its position in Fig. 2(a).

The second subsequence corresponds to the assignment of profiles to ships. For this purpose, the interval [0, 1) is equally divided into the number of possible profiles $|P_i|$ per each ship, resulting in $|P_i|$ subintervals corresponding to each profile. If an allele value is found in one of these intervals, then the corresponding profile is assigned. The position of this allele in the subsequence corresponds to the number of the ship. For a better understanding, an example is shown in Fig. 2(c), where the second subsequence of the chromosome presented in Fig. 2(a) is used in order to assign the profiles to the ships. In this example, when the first allele is considered, its corresponding ship is number 1 due to its position in the subsequence. The corresponding profile for that ship is profile 1 since the value of this allele is found in the interval corresponding to that profile. The profile assigned to ship 2 is number 2 due to the fact that the allele corresponding to the second position in the subsequence is found in the interval corresponding to the profile 2.

Once the berthing order and the assignment of profiles are known, we proceed to schedule the berthing position and time for each ship $i \in N$. For this purpose we use two scheduling methods detailed in Algorithms 2 and 3. They consist of traversing the resulting vector of berthing order, v, in order to schedule each ship in a berth taking into account its assigned quay crane profile. The application of these procedures is performed sequentially. Namely, firstly the Algorithm 2 is employed in order to obtain a feasible schedule and only in case it is unable to find a feasible schedule, the Algorithm 3 is applied. If there is no solution that satisfies the spatial, temporal and quay cranes constraints after applying both algorithms, the solution is discarded.

Algorithm 2 determines the cheapest berth j^* in terms of housekeeping cost per container (line 4). Thus, a value, $C(j)$, which takes into account the housekeeping cost among berths, is calculated for each berth j. $C(j)$ is defined as follows:

$$C(j) = \alpha_1 \cdot d_{jj} + \alpha_2 \cdot \frac{\sum_{k \in M \setminus \{j\}} d_{jk}}{M-1}, \quad \forall j \in M, \tag{2}$$

The value obtained from (2) is used in Algorithm 2 for assigning the vessels to the berths in the decoding process. The parameters α_1 and α_2 are used for having more preference for the berth with the cheapest housekeeping cost. These parameters are established in the computational experimentation. The ships are initially placed at berth j^* as long as there is enough space and quay cranes available (lines 6–8). If a ship cannot be assigned to berth j^*, it will be placed at the feasible berth with the smallest increase in housekeeping due to the interchange of containers with the ships already assigned (lines 10–20). In doing so, the value $C(i, j)$ (expression (3)) for determining the housekeeping cost considering the containers exchanged, is used

$$C(i,j) = \sum_{k \in N} \sum_{w \in M} f_{ik} d_{jw} \gamma_{kw} \tag{3}$$

Its value corresponds to the cost of assigning ship i to berth j. The γ_{kw} value is set to 1 if ship k is already assigned to berth w and 0, otherwise. This information is used in the algorithm for storing in a vector h the berths non-decreasingly ordered regarding the housekeeping costs (line 10). That is, in $h[1]$ is stored the cheapest berth in terms of housekeeping derived by the interchange of containers

between the ship to be assigned and the already assigned ships. The ship is placed in that berth as long as there is enough space and quay cranes available (lines 14–17). Otherwise, the next cheapest berth is proven (line 19). In case Algorithm 2 is unable to assign a ship to any berth (lines 20–21), the Algorithm 3 is applied.

Algorithm 3 differs from Algorithm 2 in the way it assigns the ships to berths; that is, it places each ship at the following berth with respect to the berth assigned to the previous ship. In this regard, the first ship will be assigned to berth one (line 4). The next ships are allocated in the following berth to the one where the previous ship have been assigned (lines 6–11). In case a ship cannot be placed at a berth due to fact that there is neither space nor sufficient quay cranes available (line 6), the following berth is proven (lines 12–17). The solution is discarded as long as there is no possible berth that satisfies the space, temporal and quay cranes constraints (line 18).

Algorithm 2. Berthing positioning algorithm using housekeeping information

```
1  v → Berthing order vector
2  r ← 1
3  infeasible ← FALSE
4  j* ← cheapest berth using expression (2)
5  while r ≠ N ∧ infeasible = FALSE do
6      if enough time in berth j* ∧ quay cranes available then
7          Allocate ship v[r] in berth j*
8          r ← r + 1
9      else
10         Calculate non-decreasing housekeeping cost vector h using expression (3)
11         s ← 1
12         assigned ← FALSE
13         while s ≠ M ∧ assigned = FALSE do
14             if enough time in berth h[s] ∧ quay cranes available then
15                 Allocate ship v[r] in berth h[s]
16                 r ← r + 1
17                 assigned ← TRUE
18             else
19                 s ← s + 1
20             if assigned = FALSE then
21                 infeasible ← TRUE
```

Algorithm 3. Equally berthing positioning algorithm

```
1  v → Berthing order vector
2  r ← 1
3  infeasible ← FALSE
4  s ← 1
5  while r ≠ N ∧ infeasible = FALSE do
6      if enough time in berth s ∧ quay cranes available then
7          Allocate ship v[r] in berth s
8          r ← r + 1
9          s ← s + 1
10         if s > M then
11             s ← 1
12     else if s < M then
13         s ← s + 1
14         go to step 5
15     else if s > M ∧ there are berths to be proven then
16         s ← 1
17         go to step 5
18     else if v[r] was unable to be assigned to any berth then
19         infeasible = TRUE
```

Once the decoding process is over, the complete decoded solution is evaluated according to the objective function (1). To improve the understanding of the algorithms aforementioned and complete the decoding process started in Fig. 1, a detailed explanation of the use of the algorithms is described as follows. The related data concerning this example is reported in Tables 1–4. As already mentioned, initially the allocation of ships to berths is performed by applying Algorithm 2. It firstly determines the berth j^* according to the expression (2). Considering the parameters α_1 and α_2 to be equal to 0.75 and 0.25, respectively, the cheapest berth j^* is berth 1 with a value of 12. Thus, the ship 1 is placed at berth 1 since there is sufficient space and available quay cranes. In the next iteration, given that the ship 2 is unable to be allocated at berth 1 because there is not enough berth availability, it has to be placed in another berth, which is determined by the expression (3). In this case, the berth 2 is the cheapest by means of housekeeping cost. The ship 2 is then assigned to that berth. The next ship to be assigned is number 4, the Algorithm attempts to place it at berth j^*, but since there is no available space, the ship is then placed at berth 2 by using the expression (3). Then, the ship 3 cannot be placed at berth j^* nor at berth 2, so that the Algorithm places it according to expression (3) at berth 3. The following ship, the ship 5, is placed at berth 3 according to the expression (3) due to unavailable space at berth j^*. Finally, the ship 6 cannot be allocated at berth j^*. Thus, it is assigned to berth 2 given that there is enough available space. In this case, the ship berthing time is delayed one time step with regard to its arrival time because otherwise the scheduling would incur in a quay crane infeasibility since it would be necessary 7 QCs at time step 12 to perform this schedule. The complete decoded solution and its objective function value are reported in Fig. 1 in Section *The Tactical Berth Allocation Problem*.

Stopping criterion

The stopping criterion used in this procedure is met if there has been a certain number of consecutive generations without improvement of the best-known individual of the population or a certain total number of generations has been reached.

Computational results

This section is devoted to present the computational experiments carried out in order to evaluate the effectiveness of the proposed BRKGA. It has been implemented in ANSI C and executed on a computer equipped with an Intel 3.16 GHz and 4 GB of RAM.

The problem instances used for assessing the performance of the different approaches are extracted from the related literature (see Section Parameter setting). Moreover, a benchmark suite, which aims to study different configurations of the problem, is presented in this work (see Section *Comparison with previous literature approaches*). The computational experience includes the comparison of the BRKGA with the following algorithmic methods reported in the literature for solving the TBAP:

(i) Tabu Search and Branch&Price (T^2S-BP) presented by Giallombardo et al. [13].
(ii) Branch&Price ($B\&P$) proposed by Vacca [37].
(iii) Branch&Price ($B\&P$+INIT) proposed by Vacca et al. [38] with an initialization of the master problem that uses the heuristic proposed by Giallombardo et al. [13].
(iv) Mathematical model for the TBAP implemented in IBM Ilog CPLEX [1].

Parameter setting

The parameter setting for the BRKGA has been set by considering some of the indications proposed by Gonçalves et al. [14]. We have tested different configurations of the parameters values of the BRKGA. Once its parameter values have been fixed, we have observed through the computational experience that BRKGA exhibits a similar performance for instances of the same dimensions. Hence, we have selected one representative small-, medium-, and large-sized instance for describing the parameter setting along this section.

The parameter setting for the BRKGA has been set by considering:

- G. Number of generations.
- $|Pop|$. Population size.
- $|Pop_e|$. Elite population size.
- $|Pop_m|$. Mutant population size.
- $prob_h$. Crossover rate.
- α_1, α_2. Parameters used by Algorithm 2 in the decoding process.

Fig. 3 shows the performance of BRKGA along generations in terms of objective function value and computational time for a small-, medium-, and large-sized instance, respectively. In each plot, different configurations of the size of the population are displayed. Namely, populations of 50, 100, and 200 individuals. As can be seen, when the number of generations is bigger than 100, BRKGA has converged to the best objective value found. For a population composed of 50 individuals the performance of BRKGA is worse than in those cases when using bigger populations. Concerning the computational times, as shown in the figure, the bigger the population size, the greater the computational time required. In this regard, a population of 200 individuals exhibits a similar performance than a population of 100 individuals and it requires larger computational time. Therefore, we have selected a population size, $|Pop|$, of 100 in all our experiments and a maximum number of generations, G, of 100. Moreover, in order to avoid a waste of CPU effort due to premature convergence, an additional stopping criterion of 10 generations without improvement of the best solution found is included for the computational experience.

In order to select the values of these parameters we have applied the Friedman nonparametric statistical test (Daniel [10]) to the average objective function value reported for 9 randomly selected problem instances. After performing a preliminary computational experience for a large number of combinations of parameter values, we have recognized and set the following values: G=100, $|Pop|$ = 100 and $|Pop_e|$ = 7. Moreover, for $|Pop_m|$, $prob_h$, and α_1 we have recognized the following values $|Pop_m| \in \{7, 10\}$, $prob_h \in \{0.5, 0.7\}$, $\alpha_1 \in \{0.5, 0.75\}$ and α_2 = 1 - α_1. Thus, we applied the Friedman test for each possible combination of these parameters. In doing so, each problem instance has been solved 10 times. Table 5 shows the average objective function values for each problem instance. The characteristics of the instances are shown in the columns under the heading (*Instance*) as the number of ships N, number of berths M, the number of quay crane profiles for each ship *Prof*, and the maximum number of available quay cranes Q. The p-value associated with the Friedman statistic test is greater than 0.1, so that the null hypothesis of equality of treatments is accepted at 95% and 99% confidence. Considering this, we selected the combination G = 100, $|Pop|$ = 100, $|Pop_e|$ = 7, $|Pop_m|$ = 10, $prob_h$ = 0.7, α_1 = 0.75, and α_2 = 0.25 since it presents a suitable balance between solution quality and require computational effort. Therefore, we used this parameter setting along the remainder of this computational experience.

Comparison with previous literature approaches

The literature instances used in this paper, generated by Giallombardo et al. [13], Vacca [37] and Vacca et al. [38], are based on real data provided by Medcenter Container Terminal of Gioia Tauro (Italy). The instances are divided into sets taking into account their dimensions. To distinguish the instances proposed in the different works, the instances sets that begin with letter *A* correspond to those used by Giallombardo et al. [13], whereas those used by Vacca [37] and Vacca et al. [38] begin with letter *B*.

The instances provided by Giallombardo et al. [13] were solved using CPLEX 10.2 [1] with a time limit of 2 hours for the instances sets A1 and A2, and a time limit of 3 h for A3, A4 and A5 instances as indicated by Giallombardo et al. [13]. In order to compare T^2S-BP and BRKGA, the objective values are scaled up to 100 with respect to the upper bound (UB_{MILP}) provided by CPLEX as stated below

$$S_{Obj} = \frac{obj \cdot 100}{UB_{MILP}} \tag{4}$$

A value of 100 in the scaled function value (S_{Obj}) means that the solution is certified to be optimal. In order to compare the BRKGA with the algorithmic method presented by Giallombardo et al. [13], the objective function values are scaled as explained above.

Results reported in Tables 6 and 7 show a comparison between the performance of CPLEX (using the MILP formulation described by Giallombardo et al. [13]), T^2S-BP and BRKGA. These tables are divided into groups of three instances, where the name of each group represents high (H) or low (L) traffic volume. The characteristics of the instances are shown in the columns under the heading (*Instance*) as the number of ships N, number of berths M, maximum number of available quay cranes Q, the time horizon length T_h (measured in time steps) and the maximum number of available profiles per ship *Prof*. Under the headings T^2S-BP and BRKGA are reported the scaled objective function values S_{Obj} and the required computational times *time* measured in seconds. Moreover, it is included the difference between the scaled objectives *dif*. S_{Obj} and a ratio, imp_t, between the computation times of the algorithms calculated using the following expression:

$$imp_t = \frac{(time_{T^2S-BP} - time_{BRKGA}) \cdot 100}{time_{T^2S-BP}}(\%) \tag{5}$$

The computational results reported in Tables 6 and 7 show that BRKGA exhibits a good behaviour on all the instances. BRKGA is able to provide feasible solutions in all cases, whereas CPLEX only

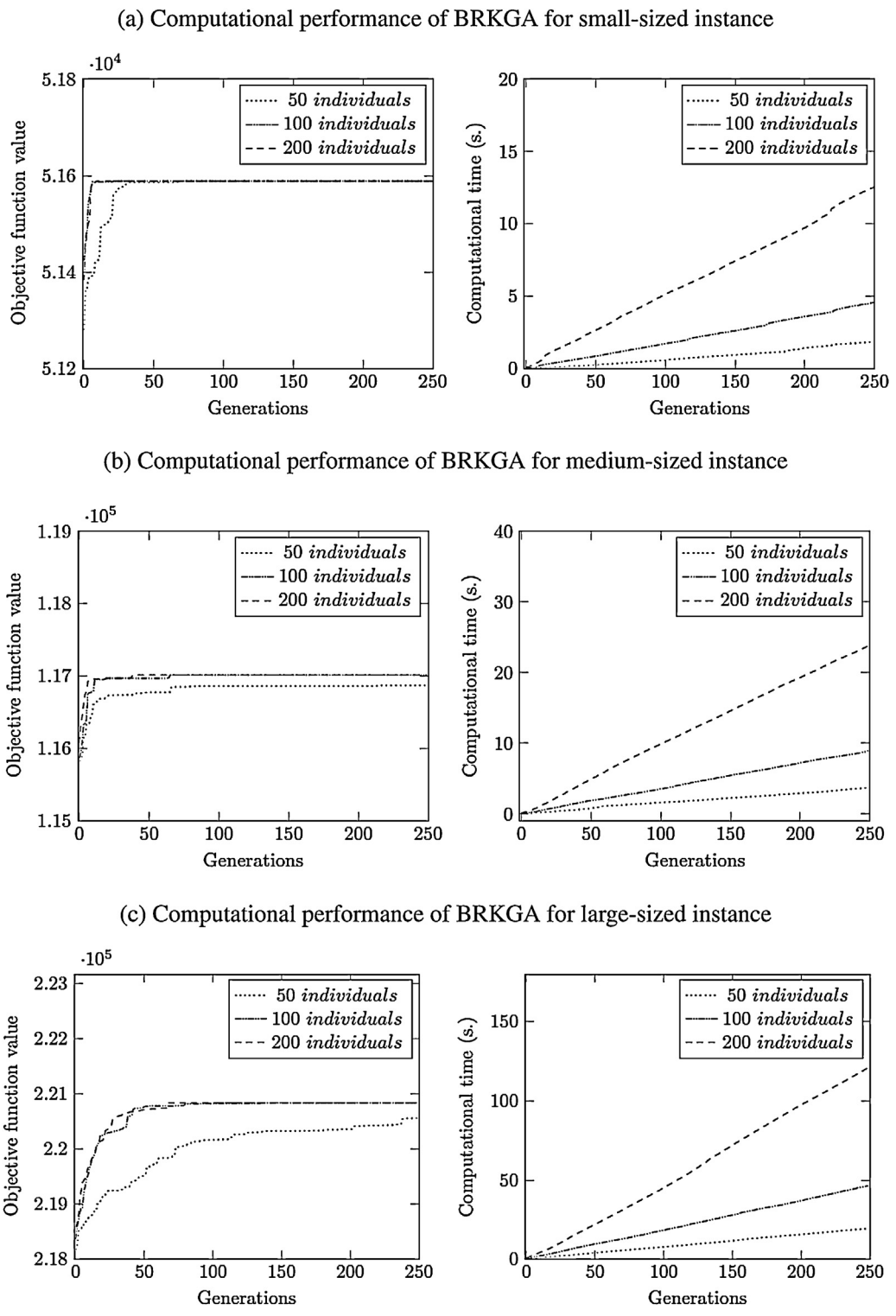

Fig. 3. BRKGA performance in terms of objective function value and computational time for different sizes of the population.

provides 8 feasible solutions and T^2S-BP does not provide feasible solutions in two cases.

BRKGA outperforms T^2S-BP. The quality of the solutions found by BRKGA is better than those provided by T^2S-BP. Moreover, the CPU time required by BRKGA is significantly smaller than the used by T^2S-BP. In the worst case the reduction is 95.37%.

It is noteworthy that as the number of ships and berths increase the required computational times by T^2S-BP and BRKGA become greater. This behaviour is also intensified when the number of profiles increases for T^2S-BP. However, as reported in the comparison with BRKGA, its temporal behaviour is not affected by variation of the number of profiles, having a similar behaviour in most groups. Only the instances set A2 presents a less uniform temporal behaviour. This is because for this set of instances, the characteristics of the scenario A1 (number of quay cranes, number of berths, time horizon) are maintained, but the number of incoming ships it is increased. Therefore the congestion and difficulty of those instances are increased. This, as shown in the

Table 5
Combination of parameter values used for the Friedman test.

Instance				$\|Pop_m\| = 7$				$\|Pop_m\| = 10$			
N	M	*Prof.*	Q	$prob_h = 0.5$		$prob_h = 0.7$		$prob_h = 0.5$		$prob_h = 0.7$	
				$\alpha_1 = 0.5$ $\alpha_2 = 0.5$	$\alpha_1 = 0.75$ $\alpha_2 = 0.25$	$\alpha_1 = 0.5$ $\alpha_2 = 0.5$	$\alpha_1 = 0.75$ $\alpha_2 = 0.25$	$\alpha_1 = 0.5$ $\alpha_2 = 0.5$	$\alpha_1 = 0.75$ $\alpha_2 = 0.25$	$\alpha_1 = 0.5$ $\alpha_2 = 0.5$	$\alpha_1 = 0.75$ $\alpha_2 = 0.25$
10	3	10	8	511,256.91	515,902.00	511,280.41	515,897.81	511,287.59	515,900.59	511,291.81	515,900.59
12	3	10	8	596,910.00	596,910.00	596,910.00	596,910.00	596,910.00	596,910.00	596,910.00	596,910.00
12	3	10	8	632,475.00	632,475.00	632,475.00	632,475.00	632,475.00	632,475.00	632,475.00	632,475.00
15	3	10	13	1,167,875.13	1,168,941.00	1,167,512.50	1,169,698.00	1,168,371.88	1,169,223.38	1,168,360.50	1,169,328.75
20	5	20	13	1,343,955.75	1,343,764.38	1,341,524.50	1,343,413.88	1,344,088.25	1,344,408.50	1,341,602.25	1,343,438.00
24	4	10	13	1,836,792.00	1,841,791.63	1,837,874.38	1,838,013.00	1,841,950.38	1,842,211.63	1,828,731.63	1,837,477.63
50	8	20	13	2,941,793.50	2,944,884.75	2,944,469.50	2,945,822.00	2,944,456.50	2,944,635.25	2,944,380.50	2,944,598.75
40	5	10	13	2,205,724.75	2,206,886.50	2,206,511.25	2,206,227.50	2,205,569.25	2,204,981.50	2,206,630.50	2,206,510.00
50	8	20	13	2,660,475.25	2,662,678.50	2,660,191.50	2,659,878.00	2,665,237.50	2,661,192.75	2,660,714.50	2,660,632.75

Table 7, does not happen in the remaining sets of instances since the time horizon is doubled. Despite the extension of the time horizon for those instances, the BRKGA still shows a relevant temporal improvement over T^2S-BP, which continues showing an increasing temporal behaviour. In this regard, T^2S-BP reports a relevant difference of execution time among instances that only differ in the number of profiles. For the group of instances L2 from A5 set, the difference between the required computational time of the instances with 10 and 20 profiles is 4823 seconds. This difference is considerably lower when applying the BRKGA, where the execution time difference is reduced to 2.59 seconds.

Tables 8–10 report a comparison among the algorithms $B\&P$, $B\&P$+INIT, the mathematical formulation implemented in CPLEX and BRKGA. Each instance belonging to Tables 8–10, is classified as high (H) or low (L) traffic volume. Under the headings $B\&P$ and $B\&P$+INIT, the objective values and the required computational times are reported. Moreover, under the headings CPLEX and BRKGA, the best objective function values and the gap with regard to the optimal solution values, opt_{sol}, are reported. The computational time required by the algorithms is reported in column t (s).

Although CPLEX is able to provide feasible solutions for several instances of B1 within the time limit, only in two cases is able to reach the optimal solution. In this regard, BRKGA is capable of providing the optimal solution for all instances, in 0.59 seconds in the worst case. Moreover, as the dimensions of the instances increase, CPLEX is unable to provide a feasible solution for most instances, being overcame by BRKGA, which achieves high quality feasible solutions in all cases. The comparison illustrated in Table 8 shows that for small-sized instances $B\&P$ is able to find the optimal solution within small computational times (27 seconds in the worst case). In this regard, the combination of $B\&P$ with an initialization method like $B\&P$+INIT requires more computational effort when it is applied to the same instances (the required computational time is about 162 seconds in the worst case).

For slightly larger instances such as B2, $B\&P$ as well as $B\&P$+INIT require a large amount of computational time in order to reach optimal solutions as reported in Table 9. For B3 instances, the computational results illustrated in Table 10 show that $B\&P$ is able to provide a feasible solution only for 2 out of the 4 instances. The $B\&P$+INIT is able to provide feasible solutions in all cases, but it

Table 6
Giallombardo et al. [13] instances computational results.

Set	Instances						CPLEX	T^2S-BP		BRKGA		Dif. S_{Obj}	Imp_t (%)
	Name	N	M	Q	T_h	*Prof.*	UB_{MILP}	S_{Obj}	t (s)	S_{Obj}	t (s)		
A1	H1	20	5	13	56	10	1,383,614	97.26	81	**97.80**	3.75	−0.54	95.37
						20	*1,384,765*[a]	97.19	172	**97.81**	3.85	−0.62	97.76
						30	*1,385,119*[a]	97.37	259	**97.82**	3.57	−0.45	98.62
	H2	20	5	13	56	10	1,474,082	97.27	82	**97.86**	3.50	−0.59	95.73
						20	*1,474,561*[a]	97.38	173	**97.86**	3.53	−0.48	97.96
						30	1,474,631	97.26	274	**97.89**	3.56	−0.63	98.70
	L1	20	5	13	56	10	1,298,356	97.30	74	**98.00**	2.56	−0.70	96.54
						20	1,298,542	97.25	158	**98.00**	3.03	−0.75	98.08
						30	1,298,542	97.06	254	**98.00**	2.76	−0.94	98.91
	L2	20	5	13	56	10	1,103,212	97.55	80	**98.06**	2.89	−0.51	96.39
						20	*1,103,516*[a]	97.39	170	**98.06**	3.20	−0.67	98.12
						30	1,103,551	97.25	295	**98.07**	1.43	−0.82	99.52
A2	H1	30	5	13	56	10	1,754,291	95.67	340	**96.74**	12.43	−1.07	96.34
						20	1,754,633	95.31	677	**96.76**	18.01	−1.45	97.34
						30	1,754,669	95.54	1009	**96.88**	37.46	−1.34	96.29
	H2	30	5	13	56	10	1,708,485	95.88	316	**96.87**	13.64	−0.99	95.68
						20	1,709,020	95.81	684	**96.84**	11.98	−1.03	98.25
						30	1,709,230	95.30	969	**96.91**	32.17	−1.61	96.68
	L1	30	5	13	56	10	1,420,485	96.55	324	**97.40**	5.90	−0.85	98.18
						20	1,420,713	96.43	652	**97.33**	5.70	−0.90	99.13
						30	1,420,819	96.18	966	**97.25**	6.20	−1.07	99.36
	L2	30	5	13	56	10	1,613,252	95.68	308	**96.82**	10.03	−1.14	96.74
						20	1,613,769	95.12	614	**96.75**	12.95	−1.63	97.89
						30	1,613,805	–	–	**96.82**	19.57	–	–
								96.57	388.30	97.44	9.32	−0.90	97.55

Bold numbers indicate the best values.
[a] CPLEX is able to provide a feasible solution.

Table 7
Giallombardo et al. [13] instances computational results.

Set	Instances						CPLEX	T^2S-BP		BRKGA		Dif. S_{Obj}	Imp_t (%)
	Name	N	M	Q	T_h	$Prof.$	UB_{MILP}	S_{Obj}	$t(s.)$	S_{Obj}	$t(s.)$		
A3	H1	40	5	13	108	10	2,553,777	97.38	1104	**97.78**	11.73	−0.40	98.94
						20	2,554,960	97.38	2234	**97.84**	11.67	−0.46	99.48
						30	2,555,422	97.25	3387	**97.75**	11.03	−0.50	99.67
	H2	40	5	13	108	10	2,561,330	97.40	1095	**97.73**	11.90	−0.33	98.91
						20	2,562,430	97.33	2198	**97.68**	11.64	−0.35	99.47
						30	2,562,644	97.27	3296	**97.64**	11.62	−0.37	99.65
	L1	40	5	13	108	10	*2,258,106*[a]	97.41	1421	**97.95**	11.12	−0.54	99.22
						20	*2,258,452*[a]	97.14	2996	**97.98**	11.45	−0.84	99.62
						30	2,258,452	96.20	4862	**97.87**	11.76	−1.67	99.76
	L2	40	5	13	108	10	2,289,660	97.41	1382	**98.02**	11.92	−0.61	99.14
						20	*2,290,662*[a]	97.34	3144	**97.94**	11.75	−0.60	99.63
						30	*2,291,301*[a]	96.60	4352	**97.90**	11.57	−1.30	99.73
A4	H1	50	8	13	108	10	2,698,239	96.52	3291	**97.42**	22.67	−0.90	99.31
						20	2,698,779	96.37	6020	**97.37**	20.70	−1.00	99.66
						30	2,698,956	96.21	9432	**97.36**	21.53	−1.15	99.77
	H2	50	8	13	108	10	3,040,930	96.03	9066	**97.22**	23.85	−1.19	99.74
						20	3,041,596	95.64	6180	**97.13**	24.28	−1.49	99.61
						30	3,041,665	95.16	9501	**97.14**	23.78	−1.98	99.75
	L1	50	8	13	108	10	2,745,446	95.97	2752	**97.31**	23.31	−1.34	99.15
						20	2,746,251	96.04	6467	**97.17**	22.21	−1.13	99.66
						30	2,746,461	95.80	9119	**97.18**	21.92	−1.38	99.76
	L2	50	8	13	108	10	2,699,412	96.18	3157	**97.46**	22.59	−1.28	99.28
						20	2,700,006	95.96	5857	**97.27**	21.87	−1.31	99.63
						30	2,700,182	96.27	8783	**97.25**	23.06	−0.98	99.74
A5	H1	60	13	13	108	10	3,227,542	95.40	6332	**96.24**	45.95	−0.84	99.27
						20	3,228,422	95.07	10,809	**96.13**	47.03	−1.06	99.56
						30	3,228,709	94.76	10,807	**96.09**	45.92	−1.33	99.58
	H2	60	13	13	108	10	3,130,833	95.54	6397	**96.17**	47.51	−0.63	99.26
						20	3,131,431	94.11	10,803	**96.16**	46.51	−2.05	99.57
						30	3,131,677	–	–	**96.16**	44.92	–	–
	L1	60	13	13	108	10	3,014,276	95.67	5807	**96.27**	45.11	−0.60	99.22
						20	3,014,877	95.40	10,803	**96.17**	44.37	−0.77	99.59
						30	3,015,054	94.45	10,806	**96.21**	43.12	−1.76	99.60
	L2	60	13	13	108	10	3,084,415	95.63	5986	**96.35**	44.41	−0.72	99.26
						20	3,085,121	95.64	10,809	**96.19**	47.00	−0.55	99.57
						30	3,085,364	95.34	10,804	**96.23**	46.42	−0.89	99.57
								96.15	6035.97	97.10	26.64	−0.98	99.49

Bold numbers indicate the best values.
[a] CPLEX is able to provide a feasible solution.

requires a large amount of computational time. The BRKGA is able to provide similar quality solutions within shorter computational times for the B1 and B2 instances and improve the quality of the solutions for the B3 instances within 4 seconds in the worst case.

As discussed above for A1–A5 instances sets, the higher the dimensions of the instances are, the greater the required computational times by solution approaches are. In the case of BRKGA, for the large-sized instances set B3, the required computational time is on average of 2.93 seconds, while for *B&P*+INIT is 9029 seconds, resulting in an improvement of nearly 9026 seconds of BRKGA over *B&P*+INIT. Moreover, concerning the variation of profiles in the same group of instances as happen in B1 set, *B&P*+INIT generally requires more computational time when the number of profiles is increased. This does not happen by applying the BRKGA or *B&P*.

Table 8
Vacca et al. [38] small-sized instances computational results.

Set	Instances						CPLEX		B&P		B&P+INIT		BRKGA		
	Name	N	M	Q	T_h	$Prof.$	$Best_{sol}$	Gap (%)	Opt_{sol}	t (s)	Opt_{sol}	t (s)	$Best_{sol}$	Gap (%)	t (s)
B1	H1	10	3	8	56	10	–	–	**790,735**	21	**790,735**	23	**790,735**	0.00	0.40
						20	–	–	**791,011**	25	**791,011**	36	**791,011**	0.00	0.62
						30	780,722	1.30	**791,045**	10	**791,045**	64	**791,045**	0.00	0.54
	H2	10	3	8	56	10	712,669	2.81	**733,276**	2	**733,276**	15	**733,276**	0.00	0.32
						20	–	–	**735,646**	7	**735,646**	29	**735,646**	0.00	0.58
						30	723,818	1.61	**735,682**	9	**735,682**	41	**735,682**	0.00	0.59
	L1	10	3	8	56	10	**515,902**	0.00	**515,902**	7	**515,902**	18	**515,902**	0.00	0.18
						20	515,991	0.40	**518,049**	5	**518,049**	29	**518,049**	0.00	0.26
						30	513,731	0.84	**518,084**	27	**518,084**	162	**518,084**	0.00	0.37
	L2	10	3	8	56	10	**564,831**	0.00	**564,831**	9	**564,831**	16	**564,831**	0.00	0.23
						20	561,504	0.60	**564,867**	7	**564,867**	37	**564,867**	0.00	0.42
						30	559,389	0.98	**564,903**	8	**564,903**	56	**564,903**	0.00	0.59
									–	11	–	44	–	–	0.43

Bold numbers indicate the best values.

Table 9
Vacca et al. [38] medium-sized instances computational results.

Set	Instances						CPLEX		B&P+INIT		B&P		BRKGA		
	Name	*N*	*M*	*Q*	*H*	*Prof.*	$Best_{sol}$	Gap (%)	Opt_{sol}	*t* (s)	Opt_{sol}	*t* (s)	$Best_{sol}$	Gap (%)	*t* (s)
B2	H1	15	3	13	56	10	–	–	**1,170,783**	1482	**1,170,783**	3507	**1,170,783**	0.00	0.54
	H2					10	1,250,124	1.74	**1,272,247**	1704	**1,272,247**	3787	1,272,236	>0.01	1.07
	L1					10	–	–	**1,098,411**	1014	**1,098,411**	1203	**1,098,411**	0.00	1.20
	L2					10	–	–	**890,211**	3583	**890,211**	8975	**890,211**	0.00	0.54
							–	–	–	1946	–	4368	–	–	0.84

Bold numbers indicate the best values.

Table 10
Vacca et al. [38] large-sized instances computational results.

Set	Instances						CPLEX		B&P+INIT		B&P		BRKGA	
	Name	*N*	*M*	*Q*	*H*	*Prof.*	$Best_{sol}$	Gap (%)	$Best_{sol}$	*t* (s)	$Best_{sol}$	*t* (s)	$Best_{sol}$	*t* (s)
B3	H1	20	5	13	56	10	–	–	1,337,077	10,800	–	10,800	**1,347,208**	2.17
	H2					10	1,221,191.00	5.12	1,429,249	10,800	–	10,800	**1,437,301**	3.89
	L1					10	–	–	1,258,150	10,800	1,256,529	10,800	**1,269,229**	2.93
	L2					10	–	–	1,070,543	10,800	1,059,231	10,800	**1,078,337**	2.71
							–	–	–	9029	–	9514	–	2.93

Bold numbers indicate the best values.

Results on new proposed instances

This subsection is devoted to present a benchmark suite for the TBAP. The new instances are based on including some realistic features, such as different housekeeping costs between berths, in contrast to the values used in the majority of the already published instances that are symmetric. This consideration is taken into account because in some ports there are different kinds of berths whose costs depend on the accessibility and handling material available at them [29]. Furthermore, different classes and subclasses of ships are considered in order to model the container cargo and exchange between ships. Therefore, there will be combinations in which heavily loaded ships will exchange their cargo with a small number of ships or low loaded ships that will exchange with several ships and so on.

For generating the proposed instances we consider the following features. The total number of ships is divided into β classes, where each class, $class_i$ ($i = 1, \ldots, \beta$) represents a group of ships. The ships within each class $class_i$ will exchange their containers with a percentage β_i of total of number of ships. Moreover, each class is divided into δ subclasses and each subclass, $subclass_j$ ($j = 1, \ldots, \delta$) will handle a percentage δ_i of the transshipment containers handled by the class it belongs to. For a better understanding, Fig. 4 illustrates an example, in which there are $\beta = 3$ classes composed by $\delta = 3$ subclasses. In this case, the ships within $class_1$ will exchange transshipment containers with the 40% of the total incoming ships, while the ships within the other classes will exchange containers with the 20% and 60% of the total incoming ships, respectively. Moreover, $class_1$ is composed of $\delta = 3$ subclasses, in which the handled transshipment containers by the ships within each subclass are 30%, 30%, and 40% respectively. In $class_2$, $subclass_1$ has a lower container workload than the other subclasses that handle a bigger load of transshipment containers.

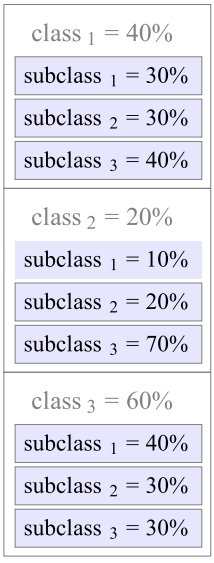

Fig. 4. Example of distribution of classes and subclasses.

In order to include ships whose stays at the container terminal are brief due to commercial/contractual matters, a parameter *TW* has been included to state that a percentage of the time windows of some ships can be reduced. Furthermore, two parameters that set the minimum and maximum number of cranes that can handle the ships, as well as, a minimum and maximum number of containers that are loaded into the ships, have also been included.

In Tables 11–13 the different parameters used for the generation of the benchmark suite are shown. A detailed description of these parameters is as follows:

- The number of incoming ships (*N*).
- The number of available berths (*M*).
- The maximum number of available quay cranes (*Q*).
- The number of quay crane profiles for each ship.
- Under the heading *Classes*, for each class *i* the percentage β_i of ships that the class will exchange containers to.
- Under the heading *Subclasses*, for each subclass *j*, the percentage δ_j of the transshipment containers that the ships of the subclass will handle.
- The percentage *TW* over the total incoming ships whose time windows have been reduced.

As can be seen in the tables, 3 sets of 5 instances with a total of 10 possible profiles per ship are generated. The construction of the profiles is performed by taking into account the cargo of the ships, the minimum and maximum number of quay cranes that can service the ships and ships time windows. Therefore, for

Table 11
Small-sized instances proposed in this work.

Instances												CPLEX				BRKGA		
Set	N	M	Q	*Prof.*	*Classes*			*Subclasses*		TW	Id							
					β_1 (%)	β_2 (%)	β_3 (%)	δ_1 (%)	δ_2 (%)			$Best_{sol}$	UB_{MILP}	S_{Obj}	t (s)	$Best_{sol}$	S_{Obj}	t (s)
											1	**596,910**	596,910	100.00	6.31	**596,910**	100.00	0.53
											2	1,020,659	1,029,695	99.12	3600	**1,026,629**	99.70	0.21
C1	12	3	8	10	10–30	30–60	60–90	50	50	20	3	–	1,490,528	–	3600	**1,446,182**	97.02	1.84
											4	–	1,377,586	–	3600	**1,370,274**	99.47	3.45
											5	–	1,415,291	–	3600	**1,370,253**	96.82	4.47
											1	**1,044,298**	1,044,298	100.00	412.92	**1,044,298**	100.00	0.45
											2	**726,295**	726,295	100.00	9.67	724,753	99.79	0.10
C2	12	3	8	10	10–30	30–60	60–90	50	50	40	3	**1,004,156**	1,004,156	100.00	792.92	**1,004,156**	100.00	0.50
											4	–	1,471,060	–	3600	**1,424,927**	96.86	8.38
											5	–	1,622,232	–	3600	**1,480,347**	91.25	17.63
											1	**1,044,828**	1,044,828	100.00	152.2	**1,044,828**	100.00	1.09
											2	**632,554**	632,554	100.00	14.89	**632,554**	100.00	0.32
C3	12	3	8	10	10–30	30–60	60–90	50	50	80	3	892,593	895,217	99.71	3600	**892,875**	99.74	0.34
											4	**823,283**	825,134	99.78	3600	797,489	96.65	0.15
											5	**1,224,061**	1,237,040	98.95	3600	1,121,967	90.70	0.15
											1	1,131,463	1,135,973	99.60	3600	**1,133,551**	99.79	0.32
											2	1,253,571	1,286,917	97.41	3600	**1,270,545**	98.73	0.90
C4	12	3	8	10	30–60	30–60	30–60	50	50	20	3	–	1,246,226	–	3600	**1,231,093**	98.79	1.01
											4	–	1,587,652	–	3600	**1,397,448**	88.02	2.11
											5	–	1,310,705	–	3600	**1,303,754**	99.47	0.62
											1	**632,475**	632,475	100.00	63.1	**632,475**	100.00	0.56
											2	1,288,261	1,293,248	99.61	3600	**1,288,930**	99.67	0.57
C5	12	3	8	10	30–60	30–60	30–60	50	50	40	3	1,078,405	1,091,851	98.77	3600	**1,086,983**	99.55	0.65
											4	1,018,914	1,046,077	97.40	3600	**1,037,292**	99.16	0.54
											5	–	1,403,550		3600	**1,360,148**	96.91	5.92
											1	772,000	772,000	100.00	24.32	**772,000**	100.00	0.21
											2	1,132,080	1,139,930	99.31	3600	**1,132,427**	99.34	1.90
C6	12	3	8	10	30–60	30–60	30–60	50	50	80	3	1,320,860	1,331,626	99.19	3600	**1,324,125**	99.44	11.68
											4	–	1,551,340	–	3600	**1,459,597**	94.09	47.95
											5	955,420	963,275	99.18	3600	**957,529**	99.40	1.36
											1	**801,207**	801,207	100.00	20.45	**801,207**	100.00	0.42
											2	1,187,046	1,207,233	98.33	3600	**1,196,880**	99.14	0.62
C7	12	3	8	10	80–90	80–90	80–90	50	50	20	3	1,203,821	1,216,988	98.92	3600	**1,210,027**	99.43	1.00
											4	–	1,350,612	–	3600	**1,337,224**	99.01	2.46
											5	–	1,388,657	–	3600	**1,359,390**	97.89	4.17
											1	990,931	993,088	99.78	3600	**993,088**	100.00	0.23
											2	1,052,430	1,063,084	99.00	3600	**1,057,124**	99.44	0.43
C8	12	3	8	10	80–90	80–90	80–90	50	50	40	3	1,076,898	1,127,080	95.55	3600	**1,118,953**	99.28	1.20
											4	–	1,618,489	–	3600	**1,304,448**	80.60	0.50
											5	–	1,462,079	–	3600	**1,327,267**	90.78	2.73
											1	**681,408**	681,408	100.00	123.98	**681,408**	100.00	0.25
											2	865,908	870,026	99.53	3600	**866,589**	99.60	0.42
C9	12	3	8	10	80–90	80–90	80–90	50	50	80	3	**1,290,837**	1,303,995	98.99	3600	1,172,727	89.93	0.19
											4	**806,467**	806,467	100.00	9.23	**806,467**	100.00	0.34
											5	**885,244**	887,050	99.80	3600	**885,244**	99.80	2.25

Table 12
Medium-sized instances proposed in this work.

Instances													CPLEX				BRKGA		
Set	N	M	Q	*Prof.*	*Classes*			*Subclasses*			TW	Id							
					β_1 (%)	β_2 (%)	β_3 (%)	δ_1 (%)	δ_2 (%)	δ_3 (%)			$Best_{sol}$	UB_{MILP}	S_{Obj}	t (s)	$Best_{sol}$	S_{Obj}	t (s)
D1	12	3	8	10	10–30	30–60	60–90	30	30	40	20	1	1,865,908	1,875,322	99.50	3600	**1,869,326**	99.68	0.20
												2	–	2,108,900	–	3600	**2,089,184**	99.07	3.06
												3	–	2,308,963	–	3600	**2,293,902**	99.35	3.93
												4	–	1,300,548	–	3600	**1,294,141**	99.51	0.26
												5	–	1,718,260	–	3600	**1,706,032**	99.29	0.42
D2	12	3	8	10	10–30	30–60	60–90	30	30	40	40	1	–	2,022,385	–	3600	**2,003,444**	99.06	19.06
												2	–	2,242,433	–	3600	**2,225,793**	99.26	1.90
												3	–	2,064,084	–	3600	**2,047,227**	99.18	1.97
												4	–	1,709,242	–	3600	**1,689,714**	98.86	1.34
												5	–	2,446,643	–	3600	**2,367,482**	96.76	1.98
D3	12	3	8	10	10–30	30–60	60–90	30	30	40	80	1	**2,134,232**	2,153,106	99.12	3600	2,133,715	99.10	1.34
												2	1,121,897	1,124,201	99.80	3600	**1,122,190**	99.82	0.67
												3	1,462,435	1,474,915	99.15	3600	**1,467,146**	99.47	2.68
												4	–	2,435,547	–	3600	**2,268,866**	93.16	0.32
												5	1,487,616	1,507,138	98.70	3600	**1,498,128**	99.40	2.34
D4	12	3	8	10	30–60	30–60	30–60	30	30	40	20	1	–	1,449,723	–	3600	**1,445,026**	99.68	0.28
												2	1,792,624	1,808,836	99.10	3600	**1,799,594**	99.49	0.54
												3	–	2,415,304	–	3600	**2,389,005**	98.91	4.71
												4	1,952,548	1,986,425	98.29	3600	**1,973,169**	99.33	0.42
												5	–	1,658,302	–	3600	**1,647,681**	99.36	0.29
D5	12	3	8	10	30–60	30–60	30–60	30	30	40	40	1	1,994,581	2,023,368	98.58	3600	**2,011,238**	99.40	0.95
												2	–	2,428,620	–	3600	**2,398,782**	98.77	6.18
												3	–	2,220,915	–	3600	**2,207,966**	99.42	11.46
												4	–	2,167,244	–	3600	**2,149,379**	99.18	1.04
												5	–	1,827,509	–	3600	**1,812,126**	99.16	0.42
D6	12	3	8	10	30–60	30–60	30–60	30	30	40	80	1	1,839,447	1,852,874	99.28	3600	**1,844,894**	99.57	1.03
												2	1,219,143	1,231,140	99.03	3600	**1,224,432**	99.46	0.39
												3	1,117,243	1,124,954	99.31	3600	**1,122,349**	99.77	0.36
												4	1,682,621	1,698,601	99.06	3600	**1,690,638**	99.53	2.28
												5	1,657,971	1,682,608	98.54	3600	**1,664,030**	98.90	6.31
D7	12	3	8	10	80–90	80–90	80–90	30	30	40	20	1	–	2,555,696	–	3600	**2,526,506**	98.86	78.23
												2	–	1,951,077	–	3600	**1,935,559**	99.20	0.32
												3	–	2,811,829	–	3600	**2,098,558**	74.63	0.65
												4	–	2,214,846	–	3600	**2,197,719**	99.23	0.98
												5	–	1,747,581	–	3600	**1,726,898**	98.82	0.46
D8	12	3	8	10	80–90	80–90	80–90	30	30	40	40	1	–	2,218,827	–	3600	**2,199,370**	99.12	6.75
												2	–	1,716,994	–	3600	**1,699,673**	98.99	0.56
												3	1,767,924	1,790,401	98.74	3600	**1,775,253**	99.15	0.50
												4	1,766,577	1,780,555	99.21	3600	**1,772,080**	99.52	0.39
												5	–	2,049,436	–	3600	**2,027,622**	98.94	0.90
D9	12	3	8	10	80–90	80–90	80–90	30	30	40	80	1	1,374,136	1,387,312	99.05	3600	**1,378,778**	99.38	2.40
												2	1,938,393	1,955,942	99.10	3600	**1,943,401**	99.36	1.29
												3	**1,967,348**	1,994,235	98.65	3600	1,930,444	96.80	0.50
												4	**1,341,435**	1,355,948	98.93	3600	1,340,230	98.84	1.96
												5	1,693,923	1,704,236	99.39	3600	**1,697,033**	99.58	12.15

Table 13
Large-sized instances proposed in this work.

Instances													CPLEX				BRKGA		
Set	N	M	Q	*Prof.*	*Classes*			*Subclasses*			TW	Id							
					β_1 (%)	β_2 (%)	β_3 (%)	δ_1 (%)	δ_2 (%)	δ_3 (%)			$Best_{sol}$	UB_{MILP}	S_{Obj}	t (s)	$Best_{sol}$	S_{Obj}	t (s)
												1	–	2,700,474	–	3600	**2,664,612**	98.67	8.50
												2	–	2,874,157	–	3600	**2,825,692**	98.31	16.09
E1	12	3	8	10	10–30	30–60	60–90	30	30	40	20	3	–	2,938,245	–	3600	**2,899,555**	98.68	46.20
												4	–	1,547,447	–	3600	**1,540,913**	99.58	0.50
												5	–	2,122,875	–	3600	**2,097,643**	98.81	1.10
												1	–	2,239,671	–	3600	**2,223,624**	99.28	1.56
												2	–	2,485,613	–	3600	**2,460,426**	98.99	3.17
E2	12	3	8	10	10–30	30–60	60–90	30	30	40	40	3	–	2,352,458	–	3600	**2,307,559**	98.09	13.29
												4	–	2,719,994	–	3600	**2,684,532**	98.70	46.60
												5	–	2,218,088	–	3600	**2,191,982**	98.82	1.81
												1	–	2,251,131	–	3600	**2,225,103**	98.84	9.48
												2	–	2,388,015	–	3600	**2,362,705**	98.94	9.12
E3	12	3	8	10	10–30	30–60	60–90	30	30	40	80	3	–	2,341,566	–	3600	**2,330,082**	99.51	17.86
												4	–	2,101,073	–	3600	**2,086,387**	99.30	1.65
												5	–	2,107,160	–	3600	**2,088,085**	99.09	2.18
												1	–	2,325,811	–	3600	**2,306,446**	99.17	1.23
												2	–	1,921,617	–	3600	**1,910,194**	99.41	1.09
E4	12	3	8	10	30–60	30–60	30–60	30	30	40	20	3	–	2,894,426	–	3600	**2,855,008**	98.64	12.95
												4	–	2,696,356	–	3600	**2,665,357**	98.85	3.07
												5	–	2,789,005	–	3600	**2,754,023**	98.75	4.18
												1	–	1,813,828	–	3600	**1,798,014**	99.13	1.08
												2	–	2,829,669	–	3600	**2,789,858**	98.59	30.62
E5	12	3	8	10	30–60	30–60	30–60	30	30	40	40	3	–	2,641,749	–	3600	**2,613,436**	98.93	5.25
												4	–	2,445,673	–	3600	**2,420,519**	98.97	2.12
												5	–	2,718,693	–	3600	**2,688,448**	98.89	8.75
												1	–	2,279,232	–	3600	**2,253,298**	98.86	11.35
												2	–	1,512,409	–	3600	**1,500,843**	99.24	1.04
E6	12	3	8	10	30–60	30–60	30–60	30	30	40	80	3	–	2,076,134	–	3600	**2,062,084**	99.32	1.71
												4	–	1,814,784	–	3600	**1,794,589**	98.89	1.42
												5	–	1,556,233	–	3600	**1,546,788**	99.39	1.07
												1	–	2,041,322	–	3600	**2,018,458**	98.88	0.92
												2	–	2,396,633	–	3600	**2,365,471**	98.70	2.14
E7	12	3	8	10	80–90	80–90	80–90	30	30	40	20	3	–	2,502,056	–	3600	**2,479,622**	99.10	2.37
												4	–	2,250,145	–	3600	**2,226,576**	98.95	1.17
												5	–	2,914,973	–	3600	**2,879,505**	98.78	5.17
												1	–	2,575,636	–	3600	**2,558,991**	99.35	2.31
												2	–	1,998,521	–	3600	**1,979,566**	99.05	1.28
E8	12	3	8	10	80–90	80–90	80–90	30	30	40	40	3	–	2,476,150	–	3600	**2,446,754**	98.81	3.37
												4	–	2,200,735	–	3600	**2,181,919**	99.15	1.03
												5	–	2,678,025	–	3600	**2,657,614**	99.24	7.69
												1	–	2,150,129	–	3600	**2,127,379**	98.94	3.85
												2	–	1,365,102	–	3600	**1,355,983**	99.33	0.73
E9	12	3	8	10	80–90	80–90	80–90	30	30	40	80	3	–	1,722,531	–	3600	**1,711,300**	99.35	1.04
												4	–	1,850,601	–	3600	**1,842,704**	99.57	1.12
												5	–	1,811,259	–	3600	**1,798,002**	99.27	2.57

the small-sized, there are 9 set of instances. The instance set C1, reflects an scenario where there are $\beta = 3$ classes composed by 4 ships each. Each class exchanges containers with 10–30%, 30–60% and 60–90% of the total number of vessels. Each class is divided into 2 subclasses of ships that will share 50% of the total amount of containers handle by their main class each. Note that the instance sets C2 and C3 have the same parameters as C1 but 40% and 80% of the ships have a reduction of their time windows, respectively.

The computational results presented in Tables 11–13 show a comparative analysis between CPLEX and BRKGA for the proposed instances in this work. Within C1–9 instances reported in Table 11, CPLEX is able to provide feasible solutions in the majority of the instances. However, it is able to reach the optimal solutions only for a few of them. From the comparative analysis with BRKGA, it can be highlighted that the proposed algorithm outperforms the quality in most instances with less computational time, 0.10 seconds in the best case. Furthermore, concerning the computational results for the medium-sized instances illustrated in Table 12, CPLEX is able to provide feasible solutions in several cases, but it is not able to find any optimal solution. Nevertheless, BRKGA finds feasible solutions for all the instances. In almost all instances where CPLEX provides a feasible solution, BRKGA is capable of improving it. Finally, regarding the large-sized instances reported in Table 13, CPLEX cannot provide a feasible solution. Nevertheless, BRKGA is able to find feasible solutions in all cases.

From the computational experiments it can be pointed out that the greater the number of ships that have hard time windows, the easier is for CPLEX to provide feasible solutions. Nevertheless, BRKGA provides feasible solutions in all cases regardless of the number of ships with restricted time windows, the configuration of transshipment containers or the ships cargo.

Conclusions

A Genetic Algorithm based on Random Keys for solving the Tactical Berth Allocation Problem is proposed in this paper. Additionally, with the aim of making a more comprehensive study of the behaviour of the proposed algorithm, a benchmark suite which considers some real-issues in maritime container terminals is presented.

From the computational experiments carried out in this paper it can be concluded that the problem is difficult to solve even for small-sized problem instances via a general-purpose solver such as CPLEX. The solution approaches proposed in the related literature provide feasible solutions with the disadvantage of requiring greater computational time as the dimensions of the instances grow. Nevertheless, BRKGA is able to provide high-quality solutions in short computational times. It also shows that it is adaptable for practical-size problems used nowadays. In this regard, it can be highlighted that the computational effort required by this solution approach is not strongly influenced by the dimensions of the instances since it provides a wide flexibility for solving different size instances in terms of computational time. This feature makes BRKGA worthy for practical cases where, on one hand, the number of ships may vary due to delays or changes in shipping routes and, on the other hand, the number of berths can change depending on how are they planned or intended to be used by the terminal; that is, in practice, depending on the received container traffic, terminal managers can dispense or establish restricted time windows for some berths in order to save port resources and reduce costs. In addition to the potential benefits of applying the BRKGA to real cases, it can also be used as a suitable tool in the negotiation between the container terminal managers and the shipping companies while services are being agreed. The terminal planners would be able to get a perspective of how the range of proposed services would affect the entire maritime terminal while the negotiations are being achieved. In this concern, the use of exact techniques, as can be seen from the computational results, requires a high computational effort which would cause a delay or slow-down in the decision-making process during the negotiations between the terminal and the shipping companies. The BRKGA is then appropriate for quickly getting all the possible services that the terminal could hold and offer to the shipping companies without compromising its performance or the contracts already agreed.

These facts justify the use of BRKGA as an algorithm for solving the TBAP in real-world contexts; particularly, in those integrated designs in which this problem appears as a subproblem and it has to be solved frequently. Therefore, the use of efficient procedures that provide near-optimal solutions within short computational times is preferable.

Acknowledgements

This work has been partially funded by the European Regional Development Fund, the Spanish Ministry of Science and Technology (projects TIN2012-32608, TIN2009-13363 and TIN2008-06872-C04-01). Eduardo Lalla Ruiz thanks the Canary Government the financial support he receives through his doctoral grant.

References

[1] User's manual for cplex. IBM Ilog CPLEX 12.1, 2009.
[2] J.C. Bean, Genetic algorithms and random keys for sequencing and optimization, RSA J. Comput. 6 (1994) 154–160.
[3] B. Beškovnik, Measuring and increasing the productivity model on maritime container terminals, Pomorstvo 22 (2) (2008) 171–183.
[4] C. Bierwirth, F. Meisel, A survey of berth allocation and quay crane scheduling problems in container terminals, Eur. J. Oper. Res. 202 (3) (2010) 615–627.
[5] C. Cheong, K. Tan, D. Liu, C. Lin, Multi-objective and prioritized berth allocation in container ports, Ann. Oper. Res. 180 (2010) 63–103.
[6] J.-F. Cordeau, M. Gaudioso, G. Laporte, L. Moccia, The service allocation problem at the gioia tauro maritime terminal, Eur. J. Oper. Res. 176 (2) (2007) 1167–1184.
[7] J.F. Cordeau, G. Laporte, P. Legato, L. Moccia, Models and tabu search heuristics for the berth-allocation problem, Transport. Sci. 39 (2005) 526–538.
[8] T.G. Crainic, K.H. Kim, Chapter 8 intermodal transportation, in: C. Barnhart, G. Laporte (Eds.), in: Transportation, Handbooks in Operations Research and Management Science, vol. 14, Elsevier, 2007, pp. 467–537.
[9] J. Dai, W. Lin, R. Moorthy, C. Teo, Berth allocation planning optimization in container terminals, in: C. Tang, C.-P. Teo, K.-K. Wei (Eds.), in: EBST Supply Chain Analysis, International Series In Operations Research & Mana, vol. 119, 2008, pp. 69–104.
[10] W.W. Daniel, Applied nonparametric statistics, PWS-Kent Publishing Company, Boston, 1990.
[11] M. Ericsson, M. Resende, P. P.M., A genetic algorithm for the weight setting problem in OSPF routing, J. Comb. Optim. 6 (2002) 299–333.
[12] C. Expósito, J. Velarde, B. Batista, J. Moreno-Vega, Estimation of distribution algorithm for the quay crane scheduling problem Nature Inspired Cooperative Strategies for Optimization (NICSO 2011), Studies in Computational Intelligence, vol. 387, Springer, Berlin/Heidelberg, 2012, pp. 183–194.
[13] G. Giallombardo, L. Moccia, M. Salani, I. Vacca, Modeling and solving the tactical berth allocation problem, Transp. Res. B: Meth. 44 (2) (2010) 232–245.
[14] J.F. Gonçalves, M. Resende, Biased random-key genetic algorithms for combinatorial optimization, J. Heuristics 17 (2011) 487–525.
[15] H.-O. Günter, K.-H. Kim, Container terminals and terminal operations, OR Spectrum 28 (2006) 437–445.
[16] P. Hansen, C. Oguz, N. Mladenovic, Variable neighborhood search for minimum cost berth allocation, Eur. J. Oper. Res. 191 (3) (2008) 636–649.
[17] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, 1975.
[18] A. Imai, H.C. Chen, E. Nishimura, S. Papadimitriou, The simultaneous berth and quay crane allocation problem, Transp. Res. Part E: Logist. Transp. Rev. 44 (5) (2008) 900–920.
[19] A. Imai, K. Nagaiwa, C.W. Tat, Efficient planning of berth allocation for container terminals in Asia, J. Adv. Transport. 31 (1) (1997) 75–94.
[20] A. Imai, E. Nishimura, S. Papadimitriou, The dynamic berth allocation problem for a container port, Transp. Res. B: Meth. 35 (4) (2001) 401–417.
[21] A. Imai, E. Nishimura, S. Papadimitriou, Berth allocation with service priority, Transp. Res. B: Meth. 37 (5) (2003) 437–457.
[22] E. Lalla-Ruiz, B. Melián-Batista, J. Moreno-Vega, Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem, Eng. Appl. Artif. Intel. 25 (6) (2012) 1132–1141.

[23] C. Liang, Y. Huang, Y. Yang, A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning, Comput. Ind. Eng. 56 (3) (2009) 1021–1028 (Intelligent Manufacturing and Logistics).
[24] A. Lim, The berth planning problem, Oper. Res. Lett. 22 (23) (1998) 105–110.
[25] M. Monaco, M. Sammarra, Berth allocation problem: a strong formulation solved by a lagrangean approach, Transport. Sci. 41 (2) (2007) 265–280.
[26] R. Moorthy, C.-P. Teo, Berth management in container terminal: the template design problem, OR Spectrum 28 (4) (2006) 495–518.
[27] K.G. Murty, J. Liu, Y. wah Wan, R. Linn, A decision support system for operations in a container terminal, Decis. Support Syst. 39 (3) (2005) 309–332.
[28] E. Nishimura, A. Imai, S. Papadimitriou, Berth allocation planning in the public berth system by genetic algorithms, Eur. J. Oper. Res. 131 (2) (2001) 282–292 (Artificial Intelligence on Transportation Systems and Science).
[29] on Trade, U. N. C., Development. Port performance indicators, 2010 http://www.unctad.org
[30] on Trade, U. N. C., Development. Review of maritime transport, 2012 http://www.unctad.org
[31] K. Park, K. Kim, Berth scheduling for container terminals by using a sub-gradient optimization technique, J. Oper. Res. Soc. 53 (9) (2002) 1054–1062, cited By (since 1996) 37.
[32] M.L. Pinedo, Scheduling: Theory, Algorithms, and Systems, 3rd ed., Springer Publishing Company, Incorporated, 2008.
[33] B. Raa, W. Dullaert, R.V. Schaeren, An enriched model for the integrated berth allocation and quay crane assignment problem, Expert Syst. Appl. 38 (11) (2011) 14136–14147.
[34] W.M. Spears, K.A.D. Jong, On the virtues of parameterized uniform crossover, in: Proceedings of the Fourth International Conference on Genetic Algorithms, 1991, pp. 230–236.
[35] R. Stahlbock, S. Vo B, Operations research at container terminals: a literature update, OR Spectrum 30 (2008) 1–52.
[36] D. Steenken, S. Vo B, R. Stahlbock, Container terminal operation and operations research – a classification and literature review, OR Spectrum 26 (2004) 3–49.
[37] I. Vacca, Container Terminal Management: Integrated Models and Large-Scale Optimization Algorithms, École Polytechnyque Fédérale de Lausanne, 2011 (Ph.D. thesis).
[38] I. Vacca, M. Salani, M. Bierlaire, An exact algorithm for the integrated planning of berth allocation and quay crane assignment, Transport. Sci. 47 (2) (2013) 148–161.
[39] I.F. Vis, R. de Koster, Transshipment of containers at a container terminal: an overview, Eur. J. Oper. Res. 147 (1) (2003) 1–16.
[40] F. Wang, A. Lim, A stochastic beam search for the berth allocation problem, Decis. Support Syst. 42 (4) (2007) 2186–2196.
[41] C. Zhang, L. Zheng, Z. Zhang, L. Shi, A.J. Armstrong, The allocation of berths and quay cranes by using a sub-gradient optimization technique, Comput. Ind. Eng. 58 (1) (2010) 40–50.