Applied Soft Computing 13 (2013) 4063–4076

Contents lists available at ScienceDirect

Applied Soft Computing

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

Hybrid Estimation of Distribution Algorithm for the Quay Crane Scheduling Problem

Christopher Expósito-Izquierdo[a,*], José Luis González-Velarde[b], Belén Melián-Batista[a], J. Marcos Moreno-Vega[a]

[a] Dpto. de Estadística, IO y Computación, ETS de Ingeniería Informática, Universidad de La Laguna, Spain
[b] Centro de Manufactura y Calidad, Tecnológico de Monterrey, Mexico

ARTICLE INFO

Article history:
Received 5 July 2012
Received in revised form 22 April 2013
Accepted 8 May 2013
Available online 2 June 2013

Keywords:
Quay Crane Scheduling Problem
Container terminal
Estimation of Distribution Algorithm
Local search
Heuristics

ABSTRACT

The competitiveness of a container terminal is highly conditioned by the time that container vessels spend on it. The proper scheduling of the quay cranes can reduce this time and allows a container terminal to be more attractive to shipping companies. The goal of the Quay Crane Scheduling Problem (QCSP) is to minimize the handling time of the available quay cranes when performing the tasks of loading and unloading containers onto/from a container vessel. This paper proposes a hybrid Estimation of Distribution Algorithm with local search to solve the QCSP. This approach includes a priori knowledge about the problem in the initialization step to reach promising regions of the search space as well as a novel restarting strategy with the aim of avoiding the premature convergence of the search. Furthermore, an approximate evaluation scheme is applied in order to reduce the computational burden. Moreover, its performance is statistically compared with the best optimization method from the literature. Numerical testing results demonstrate the high robustness and efficiency of the developed technique. Additionally, some relevant components of the scheme are individually analyzed to check their effectiveness.

1. Introduction

Nowadays, in order to cope with market demands, public institutions have built large maritime container terminals. A container terminal is a logistic point for the container exchange within an intermodal transportation network. Its main objective is the effective container transfer between different transportation modes. Typically, transportation modes in a container terminal are both maritime (container vessels) and land transportations (trucks and trains). These facilities are complex to manage due to the large number of processes that are interrelated. In the works of Vis and de Koster [39], Steenken et al. [37] and Stahlbock and Voß [36] exhaustive studies of the logistical problems as well as overviews of the most highlighted optimization methods are conducted.

There are many indicators that attempt to measure the productivity of a container terminal. Some of the most prominent are those related to the reduction of the time that container vessels spend at the container terminal, termed as *turnaround time*. The main causes that produce a large turnaround time are the high rate of utilization of its infrastructures (berths and existing equipment) in the wake of increased maritime traffic and the improper use of quay cranes that provide service to container vessels. The number of containers handled by the quay cranes gives rise to an indication of the efficiency of operators and the machinery characteristics. Thus, increasing the number of containers handled by each quay crane is an issue of interest for container terminal managers in order to counteract the increase in the capacity of container vessels. Recently, many advances have occurred in this direction. Among the major improvements in the productivity of quay cranes the incorporation of double-sided operation systems, multi-lift spreaders or quay cranes with double-trolley must be emphasized. Chao and Lin [6] present an analysis concerning these advances.

Loading and unloading operations onto/from a container vessel must be performed in response to a well-defined stowage plan. A stowage plan determines the position of each container on board following a coordinate system that establishes its bay, row and tier (bay-row-tier system). Usually, containers with the same destination port, size or weight belong to the same group and are located adjacent to each other with the aim of easing their handling operations. In this regard, a task is defined as the operations carried out according to some specific characteristic. Several definitions of a task can be found on the basis of its level of aggregation. For example, a task can include the loading or unloading operations into a defined bay or those belonging to the same container group. Therefore, a stowage plan is frequently composed of several tasks.

* Corresponding author.
E-mail addresses: cexposit@ull.es (C. Expósito-Izquierdo), gonzalez.velarde@itesm.mx (J.L. González-Velarde), mbmelian@ull.es (B. Melián-Batista), jmmoreno@ull.es (J. Marcos Moreno-Vega).

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.05.006

This paper addresses the Quay Crane Scheduling Problem (QCSP), whose goal is to define the sequence of movements to be performed by the allocated quay cranes in order to complete the loading and unloading tasks of a container vessel. It is assumed the quay cranes have similar technical characteristics. Moreover, a task is considered as the loading or unloading operation of a group of containers belonging to the same bay onto the container vessel. The objective of the problem is to minimize the overall container vessel service time (makespan).

In order to solve the QCSP, an Estimation of Distribution Algorithm (EDA) with local search (EDA/LS) is discussed. This scheme exploits a priori knowledge about the problem with the aim of reaching high-quality schedules. Additionally, a novel restarting strategy to prevent the premature convergence of the search and to guide it toward insufficiently explored regions is developed. Furthermore, to reduce the computational burden produced by the conventional evaluation procedure, a relaxed objective function is considered. Finally, an adaptive stopping criterion to determine the proper finishing of the search is presented. Computational experiments demonstrate the performance of the proposed scheme is better than those of previous related approaches and the suitability of its components. The Wilcoxon nonparametric test (Sheskin [35]) is performed to validate the relevance of the experimental results.

The main contribution of this paper is the development of a hybrid algorithm that combines an EDA with LS to solve the QCSP. This algorithm reduces significantly the computational time required by other algorithms from the literature while reaching high-quality schedules. The efficiency of the proposed algorithm supposes an important contribution to the management of container vessels due to the fact that it allows to tackle more general approaches where different logistical problems are integrated. The integration of problems will eventually lead to develop an intelligent system applicable to real situations.

The remainder of this paper is organized as follows. Section 2 presents an overview of the works concerning the QCSP and their noteworthy contributions. Section 3 provides an exhaustive description of the QCSP. Section 4 proposes a hybrid algorithm based on the application of an EDA and a LS to solve the QCSP. Section 5 discusses the computational experiments. Finally, Section 6 addresses the main concluding remarks and future lines of work.

2. Literature review

Numerous surveys have been conducted regarding the minimization of the turnaround time of a container vessel by scheduling the quay cranes at a container terminal. However, only a few studies on quay crane scheduling considering container groups of an individual container vessel and with a well-defined number of available quay cranes through its service time have been carried out.

Kim and Park [21] formulate the QCSP as a mixed integer programming model. This paper proposes a Branch and Bound algorithm based on a reduction of the search space and a lower bound of the objective function value to reach the optimal schedules. Since high computational times of the Branch and Bound in large instances limit its application in real scenarios, the authors also develop a Greedy Randomized Adaptive Search Procedure (GRASP) which allows to obtain high-quality schedules through shorter computational times.

Moccia et al. [27] perform a detailed analysis of the model proposed by Kim and Park [21] and point out there are situations in which interferences between the quay cranes can appear. The paper presents a revised mathematical formulation with the purpose of overcoming demonstrated weaknesses. Simultaneously, several families of valid inequalities are incorporated to a Branch and Cut algorithm to solve large size instances. Computational results demonstrate the good performance of the proposed Branch and Cut algorithm compared with the Branch and Bound algorithm presented by Kim and Park [21].

Sammarra et al. [32] decompose the QCSP into a routing problem to determine the tasks processed by each quay crane and a scheduling problem to calculate starting and finishing times of the tasks. A Tabu Search is then developed in order to solve the routing problem. The considered neighbourhood structure is based on using highly promising movements to reduce the makespan of the current schedule. The evaluation of the schedule reached during the search process is performed using a disjunctive graph. The performance of the Tabu Search is comparable to the Branch and Cut proposed by Moccia et al. [27].

Bierwirth and Meisel [2] disclose the incorrectness of the mathematical model proposed by Moccia et al. [27] due to the fact that it cannot detect schedules with interferences between quay cranes in all cases. A revised formulation for the QCSP is developed, where a suitable temporal distance between any two tasks is included. They present a heuristic called *Unidirectional Scheduling* (UDS) based on a tree search. UDS explores the space of unidirectional schedules; that is, schedules where the direction of movement does not change after the initial repositioning and each quay crane has the same direction of movement. UDS reaches the best-known schedules for the benchmark suite used in previous works by means of very short computational times. Tests are enlarged to check the behaviour of UDS in more complex scenarios. In every case, UDS is able to obtain high-quality schedules within a computational time limit of one hour.

Chung and Choy [9] propose a Genetic Algorithm for the QCSP. From an initial randomly generated population, several crossover and mutation operators are applied in order to reach high-quality schedules. This work tackles only a limited subset of the instances from the literature. In spite of this work is newer than the previously analyzed, its authors do not use the last mathematical formulation (proposed by Bierwirth and Meisel [2]), but it is based on the model developed by Kim and Park [21] and the modifications proposed by Moccia et al. [27] instead. Hence, some schedules reached in this work are not correct.

Legato et al. [22] present a new model for the QCSP that incorporates several practical issues as ready times and due dates for the quay cranes and crane-individual processing times. This paper develops an extension of the UDS proposed by Bierwirth and Meisel [2] and called *LTM method*. New bounds and branching criteria are presented with the goal of speeding up the search while considering the new practical aspects of the problem. Furthermore, the authors describe a Lagrangian-relaxation based approach aimed at computing lower bounds. Lastly, a Timed Petri Net (TPN) approach determines the overall container vessel service time and also the finishing time of each task and quay crane. The computational tests demonstrate the performance of this new proposal overcomes the aforementioned ones.

Meisel and Bierwirth [26] present a platform intended to compare optimization models and techniques for the QCSP through the generation of a benchmark suite. The proposed generation scheme is based on the principles of comparability, unbiasedness and reproducibility, and a set of parameters characterizing the service of a container vessel.

Expósito et al. [16] study the suitability of the EDAs when solving the QCSP. Their paper is a preliminary work where some useful general guidelines are defined with the aim of designing an optimization technique over the framework of EDAs. Unlike the present paper, their work does not include neither any hybridization scheme nor restarting strategy.

It is worth mentioning that several variants of the QCSP with practical relevance in container terminals have come up over the last few years. Most of them are related to the quay cranes availability. In this regard, Meisel [25] addresses the temporal availability of the quay cranes; that is, the quay cranes can only be used in the operations during predefined time slots. Furthermore, the works of Monaco et al. [28] and Expósito et al. [17] present optimization techniques in order to tackle a new variant of the QCSP, in which the movement of the cranes is restricted to well-defined areas of the quay. Contrary to the present paper, their pursued objective functions are based on reaching a schedule which allows to complete the operations of the vessel at hand within the given operational working shift. Finally, the QCSP at indented berths has been faced in the work of Chen et al. [7].

3. Quay Crane Scheduling Problem

The Quay Crane Scheduling Problem (QCSP) can be stated as follows. Let $\Omega = \{1, \ldots, n\}$ be a set of handling tasks (loading or unloading operations for a container group) and $Q = \{1, \ldots, q\}$ a set of quay cranes. Regarding the tasks, two dummy ones 0 and $T = n + 1$ are introduced to represent the starting and finishing times of the container vessel service, respectively. An additional task set is defined: $\overline{\Omega} = \Omega \cup \{0, T\}$. Each $t \in \Omega$ has a processing time p_t ($p_0 = p_T = 0$) and a location within the vessel l_t, expressed by a bay number. Since tasks can be located in different places within the same bay, there are tasks that have to be processed before others (Kim and Park [21]). For instance, unloading tasks have to be processed before loading tasks within the same bay. At the same time, there are tasks located in different bays that cannot be processed simultaneously because quay cranes cannot operate too close for safety reasons. Thus, let Φ denote the set of task pairs for which there is a precedence relationship and Ψ the set of task pairs that cannot be processed simultaneously. That is,

$$\Phi = \left\{(i, j) \mid i, j \in \Omega : i \text{ has to be completed before the starting of } j\right\}$$
$$\Psi = \left\{(i, j) \mid i, j \in \Omega : i \text{ and } j \text{ cannot be processed simultaneously}\right\}$$

Note that $\Phi \subseteq \Psi$. Within each container bay, precedence relationships completely determine a unique processing order for the corresponding tasks.

It is assumed the cranes have similar technical characteristics. They move with a speed of $\hat{t} > 0$ bays per time unit and have the same transshipment productivity. Each quay crane $q \in Q$ can operate after its earliest ready time r^q and is located on the bay l_0^q. In addition, the quay cranes are ordered according to their starting position, from the leftmost up to the rightmost bay. The required travel time of quay crane q from its starting position l_0^q to the bay location of a specific task l_t, where $t \in \Omega$, is denoted as $t_{0t}^q = \hat{t}|l_0^q - l_t|$. Similarly, the required travel time between two bay locations l_s and l_t, where $s, t \in \Omega$, is denoted as $t_{st}^q = \hat{t}|l_s - l_t|$. Moreover, the available quay cranes are mounted on a rail track system along the quay. Consequently, several quay cranes cannot operate at the same bay at the same time and they cannot cross each other. That is, the relative position of the quay cranes has to be kept during the service time. Furthermore, a safety distance δ (measured in bay units) has

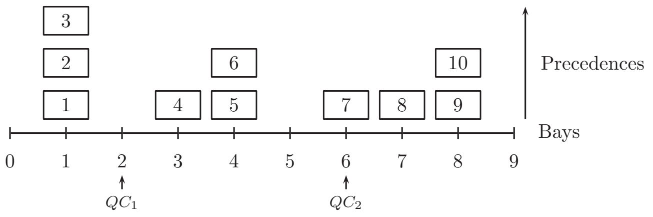


Fig. 1. Representation of the QCSP instance presented in Table 1.

to be ensured at all time in order to prevent collisions between quay cranes.

The goal of the QCSP is to determine the handling time of each task in such a way that the finishing time of the dummy task T (makespan) is minimized. The QCSP is already known to be NP-hard (Sammarra et al. [32]). A complete mathematical formulation is provided by Bierwirth and Meisel [2].

As pointed out by Bierwirth and Meisel [2], considering only unidirectional schedules for the QCSP is a widespread practice in container terminals. This policy leads to a significant reduction of the search space and, in most cases, it is possible to find an optimal schedule that respects the unidirectional behaviour. Thus, designing techniques to explore unidirectional schedules is a promising approach to obtain approximated solutions in a fast way. For this purpose, this paper proposes a hybrid Estimation of Distribution Algorithm with a local search (EDA/LS). The search is aimed at exploring the solution space composed of schedules with prespecified movement direction (left to right or right to left). With this goal in mind, the search must be applied twice in succession, varying the direction of movement of the available quay cranes in each case.

Table 1 illustrates the input data of a small size instance for the QCSP. It is composed of 10 tasks located along the container vessel and 2 quay cranes available from the starting of the service. For each task, its bay position and processing time are presented. Additionally, the precedence and non-simultaneous sets are reported. Fig. 1 depicts a graphical interpretation of the instance. Within each container bay, the corresponding tasks are represented in a bottom-up order according to the precedence relationships already defined. In this case, bays 1, 3, 4, 6, 7 and 8 have at least one task, whereas bays 2 and 5 have no tasks. Due to the precedence relationships, task 1 has to be processed before tasks 2 and 3, task 2 has to be processed before task 3, and so forth.

Following the notation introduced by Sammarra et al. [32], a schedule is denoted as $\sigma = \{\sigma^1, \ldots, \sigma^{|Q|}\}$, where σ^q refers to the sequence of tasks processed by the quay crane $q \in Q$. Additionally, σ_i^q denotes the ith task processed by q. Fig. 2 shows a unidirectional schedule for the previous example where the available quay cranes maintain the movement direction from left to right after their initial positioning. The safety distance is also kept at all times. In this case, $\sigma^1 = \{1, 2, 3, 6, 8\}$ and $\sigma^2 = \{4, 5, 7, 9, 10\}$. As can be seen, the starting and finishing time of each task are completely determined. It is noteworthy that, due to interferences between quay cranes, quay

Table 1
Input data of a QCSP instance with 10 tasks and 2 quay cranes.

Task, t	1	2	3	4	5	6	7	8	9	10
Bay position, l_t	1	1	1	3	4	4	6	7	8	8
Processing time, p_t	14	18	6	10	21	17	10	9	7	19
Precedence relationships, Φ	$\{(1, 2), (1, 3), (2, 3), (5, 6), (9, 10)\}$									
Non-simultaneous pairs, Ψ	$\{(4, 5), (4, 6), (7, 8), (8, 9), (8, 10)\} \cup \Phi$									
Quay crane, q	1	2								
Bay position, l_0^q	2	6								
Ready time, r^q	0	0								
Safety distance, δ	1									
Travel time, $\hat{t}$	1									

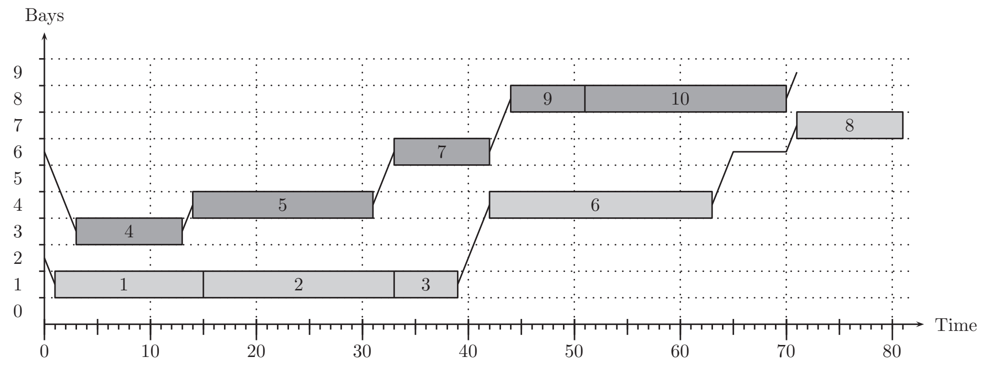


Fig. 2. A unidirectional schedule for the QCSP instance presented in Table 1

crane 1 must wait for 5 time units (idle time) to process task 8 because quay crane 2 is simultaneously processing task 10. In this example, the makespan is equal to 81 time units.

According to Sammarra et al. [32], the accurate evaluation of a schedule can be conducted through a disjunctive graph model. In this approach, the nodes correspond to each task $i \in \overline{\Omega}$, whereas the tasks performed by each quay crane $q \in Q$ constitutes a set of arcs from node 0 to node T. Furthermore, task pairs defined by precedence relationships Φ are represented by arcs. Moreover, a pair of edges represents a task pair that cannot be processed at the same time. A feasible schedule corresponds to an adequate orientation of the edges to obtain an acyclic graph. In the case of unidirectional schedules, orienting the edges from the highest index to the lowest index of quay crane produces a satisfactory orientation. The makespan of the corresponding schedule is defined by the length of the longest path in the disjunctive graph from its initial dummy task 0 to its finishing dummy task T. A detailed explanation concerning the evaluation process is developed by Bierwirth and Meisel [2]. However, in this work an approximate evaluation is considered in order to reduce the computational time required to evaluate the schedules. Section 4.6 presents an exhaustive explanation regarding to the evaluation relaxation proposed.

4. Estimation of Distribution Algorithm

Estimation of Distribution Algorithm (EDA) is a non-deterministic optimization methodology based upon populations included into the evolutionary computation field. It was first introduced by Mühlenbein and Paaß [29]. EDAs have received increasing interest within the scientific community because of the good results achieved at solving hard and complex problems in many heterogeneous knowledge areas such as feature selection (Hong et al. [15]), knapsack (Shah and Reed [34]) or dynamic problems (Yuan et al. [41]). However, scheduling has become a favorable field for the study and development of EDAs. This fact is derived from the numerous works published over the last years in this application area. Several examples are the papers by Jarboui et al. [19], Chen and Chen [8] and Wang and Fang [24].

EDAs base their behaviour on the foundations of probability theory. Along the search process, they maintain a probabilistic learning model that explicitly records statistical information about the explored search space. Moreover, such algorithms can be seen as an extension of genetic algorithms, although EDAs do not use neither crossover nor mutation operators. Instead, new solutions are generated from information gathered by the probabilistic learning model.

The main issue that differentiates EDAs from the most widely used metaheuristic algorithms is the explicit use of probabilistic learning models. Its aim is to accurately identify the remarkable features of promising candidate solutions from the population to efficiently explore the solution space. Therefore, in order to guide the search toward promising regions of the search space, the features of candidate solutions must be effectively selected by means of a proper selection procedure. Within a theoretical framework, the improvement at identifying the features of high-quality solutions of the problem leads the search, after a certain number of generations, toward the global optimum.

Probabilistic learning models have a high level of expressiveness that can represent the possible relationships between the variables of the problem to be solved. Moreover, taking into account how these interactions are treated, EDAs can be classified into different categories (Hauschild and Pelikan [13]). In the simplest approach, the problem variables are assumed to be independent from each other. Consequently, the probability distribution associated with any variable is considered in an isolated way. In a more elaborate scheme, the probabilistic learning model considers interactions between variables pairwise. Finally, EDAs can deal with multivariate interactions for solving problems with a large number of dependencies between problem variables.

In general terms, the execution of a basic EDA starts building an initial probabilistic learning model, M. This model is sampled to generate the solutions conforming the initial population, $P(g)$. At each generation g, a representative set of solutions S_e belonging to the population is selected in order to identify good features and update the model. A new population R is sampled from the model and replaces the previous one in a partial or complete way. These steps are repeated until a defined stopping criterion is fulfilled. A generic pseudocode of an EDA is depicted in Algorithm 1.

Algorithm 1. Generic pseudocode of an Estimation of Distribution Algorithm

```
$g = 0$
Initialize the probabilistic learning model, $M(g)$
$P(g)$ = Generate the initial population from $M(g)$
repeat
  $S_e$ = Select a subset of solutions from $P(g)$
  Update $M(g)$ according to solutions from $S_e$
  $R$ = Generate a new population from $M(g)$
  $P(g+1)$ = Replace solutions from $P(g)$ with solutions from $R$
  $g = g + 1$
until Stopping criterion is met
```

EDAs have a high capacity to explore the solution space and locate promising regions, but present certain shortcomings in the intensification of the solutions. One of the available options to overcome these limitations is to combine them with other optimization techniques that have a high level of exploitation. This strategy allows to keep a proper balance between diversification and intensification during the search. Numerous approaches based upon the hybridization of EDAs and other complementary techniques have been published. See the papers of Zhang et al. [43,42] and Ahn et al. [1].

In this work, a hybrid scheme that combines an EDA with a local search (EDA/LS) is developed with the aim of solving the QCSP. This scheme uses a priori knowledge about the problem in the initialization step to achieve promising regions of the search space. In addition, a restarting strategy based on the selection of a problem variable subset is included with the goal of preventing premature convergence. The restarting strategy is applied after λ generations without improvement in the best objective function value from the current population. Furthermore, an approximate evaluation approach allows to determine the objective function value of the schedules found along the search in an inexpensive way and with a small error. Finally, this scheme incorporates an adaptive stopping criterion to finish the search when a satisfactory schedule is reached. Fig. 3 depicts a flowchart showing how the different search components fit together. Its activities are numbered in order to ease the perusal of the reader through the remainder of this section. A detailed discussion about the individual components is presented in the following subsections.

4.1. Population and probabilistic learning model

The proposed scheme keeps at each generation g a population $P(g)$ with N schedules. In order to keep a high diversity level along the search, schedules belonging to the population are different from each other (Talbi [38]).

On the other hand, a probabilistic learning model $M(g)$ based on a probability matrix with $|Q|$ rows and $|\Omega|$ columns is used. Each value $m_{qt}(g)$ defines the probability of assigning the task t to the quay crane q in the model sampling step. That is,

$$M(g)=\begin{pmatrix} m_{11}(g) & m_{12}(g) & \cdots & m_{1|\Omega|}(g) \\ m_{21}(g) & m_{22}(g) & \cdots & m_{2|\Omega|}(g) \\ \vdots & \vdots & \ddots & \vdots \\ m_{|Q|1}(g) & m_{|Q|2}(g) & \cdots & m_{|Q||\Omega|}(g) \end{pmatrix}$$

The assignment of tasks to the available quay cranes is exclusive, in such a way that, the following constraint has to be satisfied

$$\sum_{q\in Q} m_{qt}(g)=1,\ \forall t\in\Omega,\ \forall g \quad (1)$$

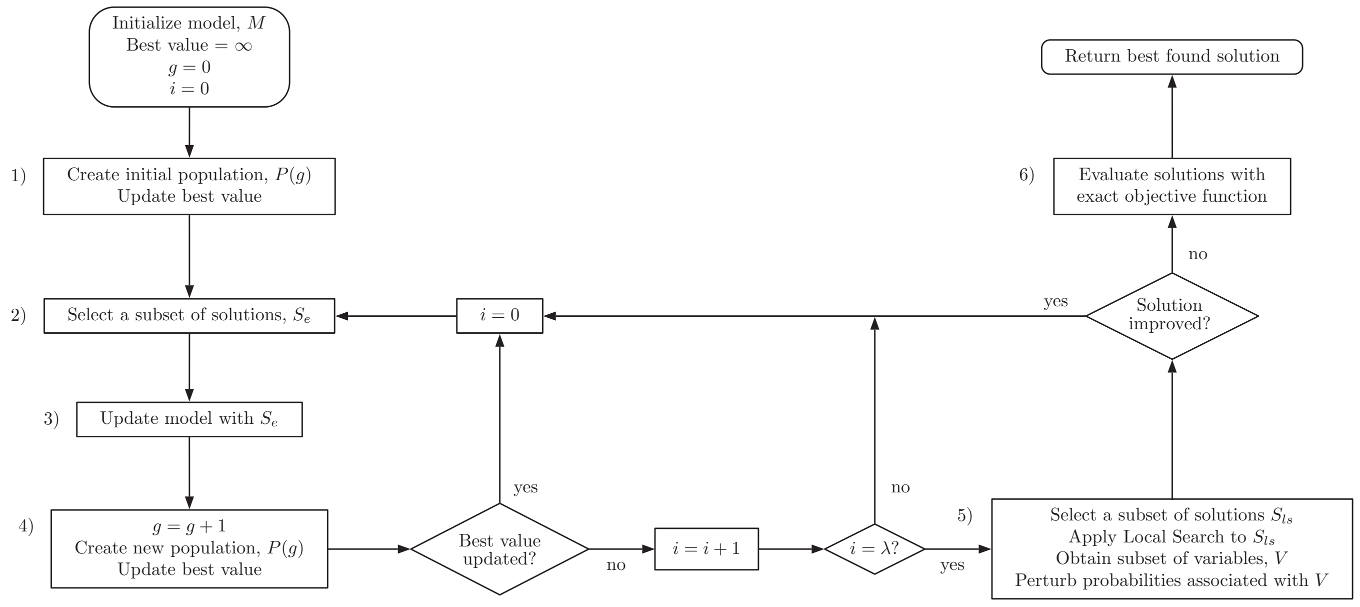
4.2. Initialization of the probabilistic learning model

Typically, there is no knowledge about the location of regions with high-quality solutions in the search space of the problem to solve. In such cases, the objective of the initial population is usually to obtain a proper diversification degree that allows a wide exploration of the search space and avoids the premature convergence of the algorithm. Different initialization strategies can be applied but, in most EDAs, the initial population is randomly created according to the uniform distribution over all feasible solutions (Zhang et al. [44]).

The availability of specific knowledge can significantly improve the performance of EDAs. There are multiple ways to include knowledge into the search process. One widely used method in evolutionary algorithms is the *population seeding* (Saavedra-Moreno et al. [31]). It is based on including a set of solutions with some defined features into the initial population in order to influence on the evolution of the search. Usually, it includes a number of high-quality solutions generated through a particular heuristic to keep a balance between the quality and the dispersion of initial solutions. Moreover, if the dependencies between variables are known, it is possible to restrict the model structure to have them taken into account from the beginning of the search (Hauschild et al. [14]).

The QCSP structure allows to incorporate a priori knowledge into the scheme of the proposed algorithm in a straightforward manner. In this case, the probabilistic learning model is initialized in such a way that it can guide the search toward regions of high influence. This step corresponds to the starting point of the flowchart depicted in Fig. 3.

In the context of QCSP, the model initialization with uniformly distributed probabilities means a task has the same probability of being processed by each one of the available quay cranes. Consequently, the uniform initialization procedure does not take into account information about the location of the tasks and the quay



Initialize model, M
Best value = ∞
$g = 0$
$i = 0$
1) Create initial population, $P(g)$
Update best value
2) Select a subset of solutions, S_e
3) Update model with S_e
4) $g = g + 1$
Create new population, $P(g)$
Update best value
Best value updated?
yes
no
$i = 0$
$i = i + 1$
$i = \lambda$?
no
yes
5) Select a subset of solutions S_{ls}
Apply Local Search to S_{ls}
Obtain subset of variables, V
Perturb probabilities associated with V
Solution improved?
yes
no
6) Evaluate solutions with exact objective function
Return best found solution

Fig. 3. Flowchart of the EDA/LS

cranes, which can be exploited to improve the performance of the developed search. Intuitively, the probability of a given task is processed by a particular quay crane in a promising schedule is highly influenced by the distance between them. That is, a task should have a higher probability of being processed by a quay crane whose initial position is close than another one at a larger distance. In this work, the initialization step is based upon using a Gaussian distribution in a discrete way for calculating initial processing probabilities. Its use is based on the fact that the most interesting probabilistic distributions are those that allow to assign a higher weight to the quay cranes initially located close to the corresponding task and it is decremented as the distance between them increases. Let l_{min} and l_{max} denote the leftmost and the rightmost container bay with at least one task, respectively. As a first step, for each possible assignment of a task $t \in \Omega$ to a quay crane $q \in Q$, a value $h(q, t)$ is calculated as

$$h(q,t) = \frac{1}{\sqrt{l_{max}-l_{min}}\sqrt{2\pi}} \exp\left\{-\frac{|l_0^q - l_t|^2}{2(l_{max}-l_{min})}\right\}$$

The normalization of $h(q, t)$ allows to obtain the corresponding initial probabilities, $m_{qt}(0)$. That is,

$$m_{qt}(0) = \frac{h(q,t)}{\sum_{q\in Q} h(q,t)}, \forall t \in \Omega, \forall q \in Q$$

The following shows the values $h(q, t)$ and the initial probabilities $m_{qt}(0)$ for the example introduced in Table 1.

$$h(q,t) = \begin{pmatrix} 0.14 & 0.14 & 0.14 & 0.14 & 0.11 & 0.11 & 0.05 & 0.03 & 0.01 & 0.01 \\ 0.03 & 0.03 & 0.03 & 0.08 & 0.11 & 0.11 & 0.15 & 0.14 & 0.11 & 0.11 \end{pmatrix}$$

$$m_{qt}(0) = \begin{pmatrix} 0.85 & 0.85 & 0.85 & 0.64 & 0.50 & 0.50 & 0.24 & 0.15 & 0.09 & 0.09 \\ 0.15 & 0.15 & 0.15 & 0.36 & 0.50 & 0.50 & 0.76 & 0.85 & 0.91 & 0.91 \end{pmatrix}$$

It should be noted that the probabilities associated with the assignments of the tasks 1, 2, 3, 4 to the quay crane 1 are significantly large. Similarly, this fact occurs in the assignments of the tasks 7, 8, 9 and 10 to the quay crane 2. Lastly, the probabilities associated with the tasks 5 and 6 are equiprobable.

4.3. Probabilistic learning model sampling

At each generation of the search a new population is generated. It is composed of a subset of schedules belonging to the previous population and new ones sampled from the model. In this case, the subset of schedules S_e with the lowest objective function value is inserted into the new population (elitism criterion). Selected schedules are defined by means of a given percentage α, whose value is set by the user. The remaining schedules are sampled from the model. Thus, at least $n_s = N - \alpha \cdot N$ are sampled at each generation (except for the initial population, where N schedules are required). This step is represented by means of the activities 1, 2 and 4 in the flowchart depicted in Fig. 3.

Sampling procedure allows to obtain new unidirectional schedules through the assignment of tasks from the probabilistic learning model. Tasks are assigned to a specific quay crane considering the container bays in which they are located (from the leftmost up to the rightmost bay). Within each container bay, the corresponding tasks are assigned following the precedence relationships. At each step, one of the available quay cranes is selected according to the roulette wheel selection based upon a pseudo-random generator. The used probabilities are specified by the probability matrix $M(g)$ at each generation g.

The search only manages unidirectional schedules. In this regard, a meticulous assignment of the task needs to be done in order to prevent violation of the movement homogeneity. To illustrate this scenario, suppose the sampling of a schedule where the quay cranes process tasks from left to right and the container bay 4 from the instance presented in Table 1. In this case, there are 2 tasks located within the bay 4: task 5 and task 6. If task 5 is assigned to quay crane 2, task 6 can be assigned either to quay crane 2 or 1. However, if task 5 is assigned to quay crane 1, task 6 cannot be assigned to quay crane 2 because the established movement direction would be violated.

Ideally, only n_s schedules should be sampled at each generation of the search. Since the population contains schedules different from each other, it is possible that some schedules must be discarded during the sampling process because they have been previously inserted into the population. Let $n(g)$ denote the ratio of schedules inserted into the current population at each generation with respect to the total number of sampled schedules, $n(g)$ will be called the *sampling ratio*. Thus, if $n(g) = 1$ all sampled schedules are inserted into the current population, whereas the lower $n(g)$ is, the larger percentage of schedules are discarded.

The sampling ratio can be seen as an indicator of the search convergence. Let γ denote a threshold set by the user to specify the maximum number of discarded schedules. If this threshold is exceeded some probabilities of the model are perturbed in order to encourage the search to find new feasible schedules. In this case, the perturbation of probabilities is carried out by setting them to a uniform value. Each time, the perturbed probabilities are those associated with a task selected at random from Ω. Previously selected tasks are not considered in the selection process. If all tasks have been selected in the same sampling process the search stops. This scenario can happen especially in problems with a short number of tasks or quay cranes since their search space is small and, therefore, it is hard to sample new schedules from it.

4.4. Updating of the probabilistic learning model

At each generation, the probabilistic learning model is updated with information of some schedules belonging to the current population with the aim of modeling the distribution of promising areas within the search space. In this case, the subset of schedules with the lowest objective function value, S_e, is used in the updating step. See activity 3 of flowchart in Fig. 3. Let x_{qt}^{σ} denote binary decision variables, where x_{qt}^{σ} is set to 1 if task t is processed by quay crane q in the schedule σ. Let $k_{qt}(g)$ be defined as follows:

$$k_{qt}(g) = m_{qt}(g-1) + \frac{1}{|S_e|}\sum_{\sigma \in S_e} x_{qt}^{\sigma}, \forall q \in Q, \forall t \in \Omega \quad (2)$$

The probabilistic learning model at generation g is calculated by means of the normalization of $k_{qt}(g)$.

In order to illustrate this process and continue with the example introduced in Section 4.2, consider a population with $N = 100$ individuals, $\alpha = 20\%$ and the initial probabilities calculated in Section 4.2. Consequently, those $|S_e| = 20$ schedules with the lowest objective function values are considered in the updating step. Let $L(g)$ be a matrix with $|Q|$ rows and $|\Omega|$ columns, where each value l_{qt} represents the number of times that task t is processed by quay crane q in one schedule $\sigma \in S_e$ at generation g. That is, $l_{qt} = \sum_{\sigma\in S_e} x_{qt}^{\sigma}$. For example, consider the following matrix for the first generation:

$$L(0) = \begin{pmatrix} 19 & 19 & 19 & 16 & 8 & 9 & 6 & 7 & 2 & 2 \\ 1 & 1 & 1 & 4 & 12 & 11 & 14 & 13 & 18 & 18 \end{pmatrix}$$

According to Eq. (2), $k_{11}(1) = 0.85 + 19/20 = 1.8$, $k_{21}(1) = 0.15 + 1/20 = 0.2$, etc. After the normalization process, $m_{11}(1) = 0.9$, $m_{21}(1) = 0.1$, etc. The probabilistic learning model is updated as follows:

$$m_{qt}(1) = \begin{pmatrix} 0.90 & 0.90 & 0.90 & 0.72 & 0.45 & 0.48 & 0.27 & 0.25 & 0.10 & 0.10 \\ 0.10 & 0.10 & 0.10 & 0.28 & 0.55 & 0.52 & 0.73 & 0.75 & 0.90 & 0.90 \end{pmatrix}$$

Along the search, some probabilities in the probabilistic learning model can be very small because of the reduced presence that the corresponding assignments might have had in the selected schedules in previous generations. Hence, to avoid these situations, a minimum probability, p_{min}, is associated with each possible assignment. The parameter value p_{min} is set by the user.

4.5. Restarting strategy

The premature convergence and the stagnation of the current population are determinant issues that need to be carefully evaluated during the design of an EDA. The reason is that it must ensure an effective exploration of the solution space to achieve the global optimum of the addressed problem. In this regard, an option to continue the exploration is to apply restarting strategies. These strategies pursue to guide the search toward insufficiently explored regions. The most straightforward strategies use none or limited explicit information concerning the state of the search. Among them, the random and the mutation restarting can be highlighted. The former generates a new population regardless of the current one, whereas the second one applies a mutation procedure to the individuals of the population (see Zhang et al. [44]). Other approaches exploit the stored information related to the search process to diversify the search in a more intensive way (see Fleurent and Glover [11]).

The proposed EDA uses a restarting strategy based upon perturbing the probabilities associated with a problem variable subset, V. Several proposals have been developed in order to select a proper subset of the problem variables (see Li et al. [23]). In this work, a local search algorithm is used to define the problem variables to perturb at each execution of the restarting phase. At the same time, the local search has a twofold purpose, as it allows to carry out the intensification of schedules from the population.

The developed local search is based on a neighbourhood structure that allows movements between different feasible unidirectional schedules. The neighbourhood function is composed by two neighbourhood structures: reassignment and interchange of tasks between the available quay cranes. Thus, the local search iteratively applies these movements until a local optimum is reached.

Given a unidirectional schedule σ and a quay crane $q' \in Q$, the reassignment move of the task σ_i^q from quay crane q to quay crane q' removes the task from the set σ^q and adds it to set $\sigma^{q'}$. Note that the task is inserted into the proper position to fulfill the corresponding direction of movement. For instance, if the following schedule for the previous example presented in Table 1 is given

$$\sigma = \{\sigma_1 = \{1, 2, 3, 6, 8\}, \sigma_2 = \{4, 5, 7, 9, 10\}\},$$

then the reassignment of the task $\sigma_4^1 = 6$ from the quay crane 1 to the quay crane 2 produces the following schedule

$$\sigma' = \{\sigma_1' = \{1, 2, 3, 8\}, \sigma_2' = \{4, 5, 6, 7, 9, 10\}\}.$$

Similarly, the interchange of the tasks σ_i^q and $\sigma_j^{q'}$ between the quay cranes q and q' is a combined movement in which, in the first place, the task σ_i^q processed by the quay crane q is assigned to the quay crane q' and then the task $\sigma_j^{q'}$ processed by the quay crane q' is assigned to the quay crane q. If the following schedule is given

$$\sigma = \{\sigma_1 = \{1, 2, 3, 6, 8\}, \sigma_2 = \{4, 5, 7, 9, 10\}\},$$

then the interchange of the task $\sigma_4^1 = 6$ from the quay crane 1 to the quay crane 2 and the task $\sigma_3^2 = 7$ from the quay crane 2 to the quay crane 1 produces the following schedule

$$\sigma' = \{\sigma_1' = \{1, 2, 3, 7, 8\}, \sigma_2' = \{4, 5, 6, 9, 10\}\}.$$

Let $reas(\sigma, \sigma_i^q, q')$ and $inter(\sigma, \sigma_i^q, \sigma_j^{q'})$ be the reassignment and interchange movements previously defined, respectively. Given a schedule σ, the following neighourhood structures can be defined as follows.

$$N_0(\sigma) = \{\sigma'/\exists \ \ \sigma_i^q, \sigma_j^{q+1} : \quad \sigma' = inter(\sigma, \sigma_i^q, \sigma_j^{q+1})\}$$

$$N_1(\sigma) = \{\sigma'/\exists \ \ \sigma_i^q : \sigma' = reas(\sigma, \sigma_i^q, q-1) \vee \sigma' = reas(\sigma, \sigma_i^q, q+1)\}$$

The selected strategy aimed at determining the next current schedule in the local search algorithm is the first improvement, in which the first neighbour that is better than the current schedule is chosen in order to replace it. To explore the solution space more broadly, those neighbour schedules that are included into the current population are discarded from the neighbourhood exploration.

The local search is a time-consuming algorithm. Therefore, this is only applied to the subset of schedules with the lowest objective function value. This subset is defined by a percentage β, whose value is set by the user.

Let S_{ls} denote the set of fittest schedules belonging to the current population determined by the parameter β after applying the local search. V is defined as the set of variables associated with tasks performed by different quay cranes in schedules from S_{ls} with respect to the best one in the population. For each task in V, the corresponding probabilities in the probabilistic learning model are uniformily set. See activity 5 of the flowchart showed in Fig. 3.

In this work, the proposed restarting strategy is applied after the execution of λ generations without improvement in the best objective function value from the current population. The value of the parameter λ is set by the user.

4.6. Evaluation relaxation

Over the last years, several strategies have been proposed in order to enhance the performance of EDAs, efficiency-enhancement techniques (EET). Parallelization, sporadic and incremental model building and evaluation relaxation are among the most prominent ones. Such techniques are usually intended to overcome the problems arising from the evaluation of the solutions found along the search (large number of solutions or high computational cost of each one) and the probabilistic learning model building process. Sastry et al. [33] provide an overview of the most widely used EETs for EDAs.

In this work, an approximate evaluation scheme of the schedules for the QCSP is carried out to reduce the computational time required by the accurate scheme proposed by Sammarra et al. [32]. This relaxation aims to use a larger number of schedules into the population by means of a similar computational time and, therefore, keep a higher level of exploration during the search than in the case of using the aforementioned accurate evaluation method (see Jim [20]).

The interferences between quay cranes impact on the objective function value as long as these are produced for that quay crane with the largest working time. The reason is that these interferences directly delay the completion of the container vessel service. The approximate relaxation scheme proposed in this work is based upon discarding the interferences between the available quay cranes from the evaluation process. That is, the approximated objective function value for a given schedule is calculated as the largest working time among all quay cranes without the influence of the possible interferences between them. The individual working time for each quay crane can be calculated from the travel times and the processing times of the assigned tasks. Consequently, the error produced through this approximation scheme is proportional to the interferences in the schedule. In this regard, if there are no

Table 2
Benchmark suite features.

	Problem instances group								
	A	B	C	D	E	F	G	H	I
Instances	10	10	10	10	10	10	10	10	10
Quay cranes	2	2	3	3	4	4	5	5	6
Tasks	10	15	20	25	30	35	40	45	50

interferences between the quay cranes, there is no error in the approximated objective function value. It is important to point out that, intuitively, there are less interferences in high-quality schedules than in low-quality ones and, therefore, this scheme can guide the search toward them.

At the end of the search, the schedules belonging to the current population are accurately evaluated according to the scheme proposed by Sammarra et al. [32]. Only the subset of schedules with the lowest objective function value, S_e, (defined by the parameter α) is considered and the best schedule is returned. This step corresponds to the activity 6 of the flowchart in Fig. 3.

4.7. Stopping criterion

The effectiveness of the local search introduced in Section 4.5 can be used as an indicator of search process convergence. The proposed stopping criterion is based upon the performance of the local search during the restarting phase. In this regard, the execution is stopped when it is not possible to improve any schedule from the selected subset of schedules.

5. Computational experiments

This section is devoted to assess the performance of the hybrid EDA proposed in this paper. It is evaluated in comparison with the most competitive approach in the published literature. Additionally, the effectiveness of the local search, the probabilistic learning model, the initialization scheme and the speed-up obtained by means of the evaluation relaxation are individually tested. The algorithm has been implemented using the Java SE 6.0 language and all experiments have been carried out on a PC equipped with Ubuntu 10.04, a processor Intel Core 2 Duo 3.16 GHz and 4 GB of RAM.

The benchmark suite tackled along this section was initially introduced by Kim and Park [21] and later extended by Bierwirth and Meisel [2]. It is composed of 90 problem instances generated at random consecutively named from *k13* up to *k102* and grouped into 9 groups (groups A to I) with 10 instances in each one. The instance sizes comprise a wide range of conventional scenarios. The benchmark suite contains instances with a number of tasks ranging from 5 up to 50, whereas the number of quay cranes ranges from 2 up to 6. All the quay cranes are available from the beginning of the container vessel service ($r^k = 0, \forall q \in Q$) and have to keep a safety distance of one bay ($\delta = 1$). In addition, the quay cranes move with a similar speed of one bay per time unit ($\hat{t} = 1$). The previous studies consider the makespan of the reached schedules weighted with a factor of 3. This work also assumes it for comparison purposes. The benchmark suite features are depicted in Table 2.

Most previous studies are limited to instances from *k13* up to *k49* (Kim and Park [21], Moccia et al. [27], Sammarra et al. [32] and Chung and Choy [9]). In most cases, the computational times used are certainly high, as well as the quality of the reached solutions presents significant gaps with respect to the optimal solutions. Furthermore, these proposals do not ensure the interference constraints between quay cranes in all cases, in such a way that, the feasibility of the solutions is not guaranteed (see Bierwirth and Meisel [2]).

The most competitive algorithm for solving the QCSP is the LTM method (Legato et al. [22]). This optimization method is an improvement of the previous UDS heuristic (Bierwirth and Meisel [2]). It allows to reach optimal unidirectional solutions. The execution of the LTM is proposed with a stopping criterion based on a maximum execution time of 1 hour or after having inspected 100 million nodes of the search tree without improvement in the quality of the best solution. The original paper only reports results for instances from *k43* up to *k102* due to the fact that the UDS can obtain all optimal unidirectional schedules with a very competitive performance (less than 1 second) for instances from *k13* up to *k42*. Similarly, the UDS heuristic is proposed with a stopping criterion based on a maximum execution time of 1 hour. The published results for the LTM show some fluctuations concerning the computational time required in large size instances with similar features. Thus, within the same group, there are instances solved in a few seconds and other ones that require a large amount of time (for example, group E). Furthermore, the search space exploration for 6 instances is ended because the maximum number of explored nodes is reached. Moreover, the LTM presents a manifest growth in computational times as the size of instances increases in such a way that, in the largest instances (group I), there are several instances solved with an amount of computational time larger than half an hour. Consequently, the irregular performance of the LTM can decline its use in practical scenarios, where the robustness of the optimization technique must be ensured.

The results presented in this section for instances from *k13* up to *k42* are those reported by Bierwirth and Meisel [2] and obtained by means of the UDS heuristic. On the other hand, those results for instances from *k43* up to *k102* are reported by Legato et al. [22] and obtained with the LTM method. Both optimization techniques have been implemented in Java. The UDS heuristic has been executed on a PC P4 2.8 GHz, whereas the LTM method has been tested on a 3.07 GHz Intel T9900 mobile computer. Finally, the results presented for the developed hybrid algorithm correspond to the best objective function value and the average computational time in 10 executions in each case.

5.1. Parameter setting

The stochastic behaviour of EDAs gives rise to the fact that a proper selection of parameter values should be conducted. In this regard, the goal of this first experiment is to determine the best combination of parameter values for the proposed EDA. According to the general guidelines of the *Design of Experiments* (Box et al. [5]), a preliminary step in the parameter setting is to define the *factors* and *treatments*. The factors constitute the parameters of the algorithm whose values need to be determined, whereas the treatments are the candidate parameter values. In this case, the EDA uses several parameters: a population size *N*, a percentage of solutions to update the probabilistic learning model α, a percentage of solutions to apply the restarting strategy β, a threshold to check the sampling ratio γ, a number of generations without improvement after which the restarting strategy is applied λ, and a residual value for the probabilities in the probabilistic learning model p_{min}. Table 3 shows reasonable parameter values to assess during the EDA parameter setting.

Multitude of methodologies aimed at setting the best parameter values for a given optimization technique have been proposed in the literature. As pointed out by Talbi [38], generally speaking there are two different approaches for parameter setting: the off-line and the online strategies. The former sets the parameter values before starting the algorithm execution, whereas the latter one changes the parameters values during the algorithm execution. In this case, an off-line full factorial strategy is conducted in order to determine the best combination of parameter values (Ridge and

Table 3
Parameter values used in the parameter setting of the EDA.

Parameter	Value
N	$\in\{100, 250, 500, 1000, 2000\}$
α	$\in\{10, 20, 30, 40\}\%$
β	$\in\{10, 20, 30, 40\}\%$
γ	$\in\{N/2, N\}$
λ	$\in\{20, 40\}$
p_{min}	$\in\left\{\frac{1}{\lvert Q\rvert\cdot\lvert\Omega\rvert}, \frac{1}{\lvert Q\rvert}\right\}$

Kudenko [30]). That is, the EDA is executed with all the combinations of parameter values shown in Table 3. The interested reader is referred to the book by Birattari [3] for an in-depth discussion about parameter setting in metaheuristics.

Over the last years, a great deal of research has been concentrated on evaluating the influence of parameters on the performance of algorithms through statistical analysis. An interesting approach for this purpose is based upon determining the relative performance of the algorithm to evaluate under different combinations of parameter values and, afterwards, selecting the best of them (García and Pelta [12]). In this work, once the performance of the EDA with each combination of parameter values is known, the Friedman nonparametric statistical test (Daniel [10]) is used in order to state an order of performances. In the cases in which the null hypothesis of equality of treatments is rejected, the multiple comparisons test of Friedman is used with the aim of determining the differences among combinations. This approach has been already successfully applied in several works. Xu et al. [40] use the Friedman test with the goal of establishing the parameter influence on the overall performance of a Tabu Search for solving a telecommunications network design problem. Expósito et al. [18] state by the Friedman test the best parameter values of a domain-specific heuristic for the Pre-Marshalling Problem, a container rehandling problem found on the yard of container terminals. Finally, several algorithms based on statistical tests have been designed for the automatic algorithm configuration (Birattari et al. [4]).

According to the aforementioned discussion, the Friedman test is applied to the average objective function value of the EDA in the instances introduced in Table 2 with the combinations of parameter values reported in Table 3. In this case, the Friedman test at $\alpha_{friedman} = 0.05$ significance level for the objective function values indicates that there are statistically significant differences among the combinations of parameter values. In this case, the combinations with $N \in \{1000, 2000\}$, $\alpha \in \{20, 30\}$, $\beta \in \{10, 20, 30\}$, $\gamma \in \{N/2, N\}$, $\lambda = 20$ and $p_{min} = \frac{1}{\lvert Q\rvert\cdot\lvert\Omega\rvert}$ are those with the best performance. For this experimentation, the combination $N = 2000$, $\alpha = \beta = 20\%$, $\gamma = N/2$, $\lambda = 20$ and $p_{min} = \frac{1}{\lvert Q\rvert\cdot\lvert\Omega\rvert}$ is selected due to the fact that it presents an adequate balance between effectiveness and computational times.

The study of the parameter values also allows to provide a general discussion focused on their individual influence on the overall performance of the EDA:

- The population size, N, must be sufficiently large to keep a high exploration degree during the search. For small size instances, population sizes around 250 are suitable, whereas it is necessary to consider population sizes over 1000 for large size instances. It is worth mentioning that the larger the population size is, the larger the computational times.
- The value of α must allow to capture the good features of schedules from the population. Small values mean that only a reduced subset of high-quality schedules from the current population states the probabilities of the probabilistic learning model and, therefore, it might lead to a fast convergence of the search. On the other hand, large values prevent good features from having a relevant role in the updating process of the model, which can avoid to identify promising regions of the search space.
- The value of β must be selected in such a way that the local search is only applied to those schedules found in promising regions of the search space. In this regard, small values could leave promising schedules aside from the exploitation process. However, large values produce a great increase in the computational times. Therefore, in a natural way, its value should be related to the percentage of schedules involved in the updating of the probabilistic learning model.
- The value of γ should be related to the population size due to the fact that it provides an indicator of the capacity to generate new schedules to be included into it. It is straightforward to see that only those scenarios with a reduced number of feasible schedules and those in which the probabilistic learning model has converged the parameter γ has influence.
- The value of λ must allow to detect the stagnation of the search and avoid the execution of generations without improvement in the objective function value of the best solution found. Values around 20 are suitable in most of the cases.
- The parameter p_{min} prevents some assignments of tasks to quay cranes from being unfeasible in the generation process. In this regard, extremely small values could have any practical influence on the generation of schedules in large scenarios, whereas large values avoid the convergence of the search. One intuitive strategy is to choose the value of p_{min} according to the dimension of the instance to solve; for instance, as it was done in the present parameter setting.

5.2. Comparison with previous approaches

In the following, a comparison between the LTM method (or the UDS heuristic for small size instances) and the proposed algorithm is presented. Table 4 shows the computational results obtained through the proposed algorithm with respect to the UDS when solving small instances. Parameter values for EDA are those set in the previous subsection. First two columns (*Group* and *Instance*) present the group and the instance to solve. Consecutively, next columns provide the results obtained by means of the UDS heuristic and the EDA. In each case, the best objective function value (f_{UDS} and f_{EDA}, respectively) and the computational time (t_{UDS} and t_{EDA}, respectively), measured in minutes, are presented. Column *Gap* shows the relative gap between both methods ($Gap = (f_{EDA} - f_{UDS})/(f_{UDS}) \times 100$). Additionally, for each group of instances, average values are shown. In the case of the UDS heuristic and small size instances, only average computational times are reported in the original paper.

The presented results show that, for small instances, the hybrid EDA obtains the optimal unidirectional solutions in all cases ($Gap = 0.00\%$). The search space in such instances is not overly broad, in such a way that, exhaustive explorations as it is done by the UDS are really advisable due to the fact that they are capable of reaching the optimal unidirectional solution through short times. However, the proposal made in this work also presents a very competitive performance, allowing to reach the optimal unidirectional solution in the worst cases in approximately 9 s.

On the other hand, Table 5 reports the comparison between the LTM and the hybrid EDA for large size instances. In this case, the column *LB* shows a lower bound proposed by Legato et al. [22]. Columns f_{LTM} and t_{LTM} provide the makespan of the schedules delivered by the LTM method and the computational times required by this method, respectively.

In most cases, the optimal unidirectional solution is achieved by the EDA, whereas in those where it has not been reached, the gap is very small. In the worst case there is a gap of 1.03% (instance

Table 4
Comparison between UDS and hybrid EDA. Small instances.

		UDS		EDA/LS		
Group	Instance	f_{UDS}	$t_{UDS}(m)$	f_{EDA}	$t_{EDA}(m)$	Gap (%)
A	k13	453	–	453	1.29E−3	0.00
	k14	546	–	546	1.46E−3	0.00
	k15	513	–	513	1.28E−3	0.00
	k16	312	–	312	9.38E−4	0.00
	k17	453	–	453	1.41E−3	0.00
	k18	375	–	375	1.19E−3	0.00
	k19	543	–	543	1.38E−3	0.00
	k20	399	–	399	1.50E−3	0.00
	k21	465	–	465	1.20E−3	0.00
	k22	540	–	540	2.12E−3	0.00
	Avg.	**459.9**	**1.12E−5**	**459.9**	**1.38E−3**	**0.00**
B	k23	576	–	576	4.53E−3	0.00
	k24	666	–	666	6.48E−3	0.00
	k25	738	–	738	4.18E−3	0.00
	k26	639	–	639	5.47E−3	0.00
	k27	657	–	657	4.89E−3	0.00
	k28	531	–	531	4.52E−3	0.00
	k29	807	–	807	5.22E−3	0.00
	k30	891	–	891	3.74E−3	0.00
	k31	570	–	570	4.32E−3	0.00
	k32	591	–	591	6.17E−3	0.00
	Avg.	**666.6**	**3.68E−5**	**666.6**	**4.95E−3**	**0.00**
C	k33	603	–	603	1.52E−1	0.00
	k34	717	–	717	1.54E−1	0.00
	k35	684	–	684	7.46E−2	0.00
	k36	678	–	678	1.27E−1	0.00
	k37	510	–	510	6.82E−2	0.00
	k38	618	–	618	1.11E−1	0.00
	k39	513	–	513	1.09E−1	0.00
	k40	564	–	564	1.19E−1	0.00
	k41	588	–	588	1.11E−1	0.00
	k42	573	–	573	1.05E−1	0.00
	Avg.	**604.8**	**6.26E−4**	**604.8**	**1.13E−1**	**0.00**

k100). Furthermore, performing an analysis by groups, the results show that the largest gap occurs for instances from group I, where 0.07% of average gap is reached. It is important to highlight the performance on the largest instances (group I). In this case, the proposal improves the objective function value presented by the LTM method in 1 instance (*k101*), showing an improvement of 0.37%. Keep in mind that, for those instances where the execution of the LTM method was finished after reaching the maximum number of nodes without improvement in the quality of the best found solution, the optimality of the delivered solution is not guaranteed. That is, instances *k53*, *k94*, *k95*, *k97*, *k99* and *k101*. Therefore, the overall average quality of solutions achieved is only 0.018% higher compared to those presented by the LTM method. In order to analyze in a statistical manner the quality of the delivered solutions, the Wilcoxon Rank Sum test has been conducted. This nonparametric test with 95% of confidence has revealed that there is no statistical significant difference between the quality of the solutions obtained with the LTM method and the proposed hybrid EDA.

On the other hand, if an analysis about the computational times is carried out, as already noted, increasing the size of the instances gives rise to a great degradation in the performance of the LTM method. This situation is evidenced by the fact that, for many instances, the computational time largely exceeds 10 minutes. In these cases, the average computational time achieved by the hybrid EDA is noticeably lower than that of the LTM method. The maximum average computational time is about 77 s (group I), which demonstrates its high efficiency. According to the Wilcoxon Rank Sum test with 95% of confidence, there is a statistically significant difference between the computational times required by both optimization techniques. This means that the developed hybrid EDA is more efficient than the LTM method. Consequently, the hybrid EDA is suitable to be applied in real situations due to its effectiveness, efficiency and high robustness in different scenarios.

5.3. Effectiveness of the local search

The main goal of the local search is to intensify solutions from promising regions found along the search. This subsection conducts a computational test to assess its performance into the structure of the proposed algorithm.

Table 6 reports the results obtained by means of the execution of the EDA/LS and a basic EDA. Both optimization techniques use the parameter values previously set in Section 5.2. However, due to the fact that the proposed EDA/LS is based upon an implicit stopping criterion, the EDA is executed with a maximum number of generations equal to 100 for comparison purposes. This value is the conventional number of generations carried out by the EDA/LS. Column *Group* represents the instance group to evaluate. Column f_{LTM} shows the average objective function value provided by the LTM method. Next columns show the average gaps (*Gap*) with respect to the results obtained by the LTM method and the average computational times (t), measured in seconds, required by the EDA/LS and the EDA, respectively. The last row presents average values.

According to the results summarized in Table 6, several conclusions can be drawn. The EDA without local search provides competitive solutions for the QCSP that can be suitable in a large number of contexts. For instance, the optimal undirectional schedules are achieved for the small size instances in every case (groups A to D). However, as the size of the instance increases, its performance is progressively decreased and fails to achieve the optimal unidirectional schedule. That is, a gap over 3% on average is obtained for the largest instances. Due to the small computational

Table 5
Comparison between LTM and hybrid EDA. Large instances.

Group	Instance	LB	LTM		EDA/LS		
			f_{LTM}	$t_{LTM}(m)$	f_{EDA}	$t_{EDA}(m)$	Gap (%)
D	k43	860	876	0.21	876	0.21	0.00
	k44	821	822	0.20	822	0.19	0.00
	k45	825	834	0.18	834	0.14	0.00
	k46	690	690	0.19	690	0.16	0.00
	k47	792	792	0.17	792	0.17	0.00
	k48	629	639	0.19	639	0.14	0.00
	k49	880	894	0.18	894	0.22	0.00
	k50	727	741	0.17	741	0.28	0.00
	k51	778	798	0.17	798	0.20	0.00
	k52	942	960	0.17	960	0.22	0.00
	Avg.	**794.4**	**804.6**	**0,18**	**804.6**	**0.19**	**0,00**
E	k53	660	717	18.02[a]	717	0.29	0.00
	k54	756	774	0.30	774	0.33	0.00
	k55	667	684	0.30	684	0.32	0.00
	k56	670	690	0.38	690	0.31	0.00
	k57	684	705	0.31	705	0.18	0.00
	k58	769	786	0.32	786	0.33	0.00
	k59	669	687	0.29	687	0.27	0.00
	k60	766	783	0.32	783	0.23	0.00
	k61	620	639	0.30	639	0.23	0.00
	k62	831	837	0.30	837	0.39	0.00
	Avg.	**709.2**	**730.2**	**2.08**	**730.2**	**0.29**	**0.00**
F	k63	930	948	0.56	948	0.28	0.00
	k64	718	741	0.51	741	0.45	0.00
	k65	821	837	0.70	837	0.55	0.00
	k66	910	924	0.38	924	0.44	0.00
	k67	861	882	0.39	882	0.35	0.00
	k68	951	963	0.35	963	0.42	0.00
	k69	787	807	0.56	807	0.24	0.00
	k70	939	957	0.44	957	0.21	0.00
	k71	814	834	0.99	834	0.74	0.00
	k72	727	744	0.38	744	0.41	0.00
	Avg.	**845.8**	**863.7**	**0.53**	**863.7**	**0.41**	**0.00**
G	k73	840	870	6.82	870	0.30	0.00
	k74	825	843	1.56	843	0.36	0.00
	k75	660	675	0.90	675	0.64	0.00
	k76	830	852	0.81	852	0.50	0.00
	k77	679	699	0.86	702	0.53	0.43
	k78	623	642	1.45	642	0.44	0.00
	k79	722	744	0.91	744	1.05	0.00
	k80	724	750	0.88	750	0.38	0.00
	k81	711	738	0.93	738	0.26	0.00
	k82	701	717	0.91	717	0.80	0.00
	Avg.	**731.5**	**753.0**	**1.60**	**753.3**	**0.53**	**0.04**
H	k83	928	948	1.02	948	0.68	0.00
	k84	882	897	0.98	897	0.68	0.00
	k85	952	972	1.08	972	0.46	0.00
	k86	791	816	5.88	819	0.53	0.37
	k87	846	867	16.29	867	1.21	0.00
	k88	747	768	1.89	768	0.34	0.00
	k89	825	843	1.09	843	0.64	0.00
	k90	1029	1053	2.65	1053	1.42	0.00
	k91	814	837	1.83	837	0.76	0.00
	k92	876	897	2.78	897	0.79	0.00
	Avg.	**869**	**889.8**	**3.55**	**890.1**	**0.75**	**0.03**
I	k93	792	810	8.43	810	1.35	0.00
	k94	768	786	19.84[a]	786	1.66	0.00
	k95	806	834	18.87[a]	834	1.68	0.00
	k96	783	801	3.66	801	1.14	0.00
	k97	693	717	33.70[a]	717	1.32	0.00
	k98	717	735	1.52	735	1.08	0.00
	k99	823	852	20.69[a]	852	0.72	0.00
	k100	858	876	2.84	885	1.68	1.03
	k101	771	813	39.48[a]	**810**	1.11	**-0.37**
	k102	874	897	5.62	897	1.13	0.00
	Avg.	**788.5**	**812.1**	**15.45**	**812.7**	**1.29**	**0.07**

[a] Stopped at max. nodes.

Table 6
Comparison between EDA/LS and EDA.

Group	f_{LTM}	EDA/LS Gap (%)	$t(s)$	EDA Gap (%)	$t(s)$
A	459.9	0.00	0.083	0.00	0.093
B	666.6	0.00	0.297	0.00	0.297
C	604.8	0.00	6.781	0.00	3.772
D	804.6	0.00	11.534	0.00	4.657
E	730.2	0.00	17.325	0.29	4.948
F	863.7	0.00	24.527	0.49	5.753
G	753.3	0.04	31.552	1.67	6.513
H	890.1	0.03	45.017	1.58	7.329
I	812.1	0.07	77.217	3.21	8.484
Avg.:	**731.6**	**0.018**	**23.815**	**0.898**	**4.650**

burden required by the EDA (below 10 s in the worst case), it can be used to provide upper bounds in a fast way or be integrated into combined solution schemes of logistical problems.

In spite of the good balance between effectiveness and computational times reported by the EDA without local search, there are many environments where the quality of the schedule is the ultimate goal at serving large container vessels. Typically, for this type of vessels the turnaround time is several days. Therefore, reducing the time required to complete the transhipment operations associated with a container vessel involves its earlier departure time and a greater flow of transported containers around the container terminal. In this regard, incorporating the local search into the scheme of the EDA provides near optimal unidirectional schedules in every case at the expense of increasing the computational times, over 77 s on average for the largest instances. This fact can be translated as a large reduction in the handling time of container vessels compared with the handling times provided by the EDA without the local search. The reason of the improvement is that, although EDA allows to find promising regions of the search space, its ability to detect local optima is certainly limited.

5.4. Effectiveness of the probabilistic learning model

The probabilistic learning model has to provide a great capability to gather statistical information about those regions of the search space with high-quality solutions. At the same time, the solutions sampled from the model must reasonably cover all the search space in order to prevent the stagnation of the process. Therefore, its use in an optimization technique must be justified by means of a better performance than its memoryless counterparts.

This subsection pursues to assess the impact of including the proposed probabilistic learning model on the performance of the presented scheme. For this purpose, a comparison between the EDA/LS with a memoryless multistart strategy is carried out. Algorithm 2 presents the pseudocode of the multistart used in the comparison. As can be seen, in a first step a solution is generated at random and later is improved by means of the local search described in Section 4.5. These steps are repeated for *max* iterations, where *max* is a parameter whose value is set by the user.

Algorithm 2. Pseudocode of the multistart process

```
Best solution = $\infty$
for $i = 1 \rightarrow max$ do
  $S \leftarrow$ Generate random solution
  $S' \leftarrow$ Apply local search to $S$
  Update best solution found
end for
```

Table 7 summarizes the results reported by the EDA/LS and the multistart strategy. The first column (*Group*) represents the instance group at hand from the benchmark suite. The second column (f_{LTM}) shows the average objective function value reported by the LTM method at solving the corresponding problem instance group. The column *EDA/LS* shows the results obtained by means of the algorithm presented in this work. These results are the same as those previously reported in Section 5.2. Next columns show the performance of the multistart strategy with different maximum number of iterations (parameter *max*), from 1000 up to 10,000. For each algorithm, average gaps (*Gap*) with respect to the results obtained by the LTM method and computational times (t), measured in seconds, are provided. The last row shows average values.

The results presented in Table 7 disclose that the performance of EDA/LS overcomes significantly the considered multistart strategy. In spite of presenting reasonable computational times (below 2 minutes for the largest instances and $max = 10,000$), the obtained gap is clearly unacceptable for application in real environments (over *35%* on average for the largest instances). This gap is largely over *50%* for several instances.

As indicated by the results, the diversification provided by the multistart approach is poor to overcome the local optima found along the search. In this regard, considering proposals dedicated to gather knowledge about the search space in a dynamic way into the optimization technique, as it is done by means of the probabilistic learning model, allows to explore it thoroughly and increase its effectiveness.

5.5. Initialization scheme

A comparison between the proposed approach and a classic initialization scheme is developed in order to analyze the individual impact of the initialization step on the effectiveness of the EDA. As pointed out in Section 4.2, the most widespread initialization procedure of populations in evolutionary algorithms is based on the random sampling of the search space. In the context of the QCSP, a uniform distribution of probabilities among the feasible assignments is set.

Table 7
Comparison between the EDA/LS and the multistart strategy.

Group	f_{LTM}	EDA/LS		MS		MS		MS	
				$max = 1000$		$max = 5000$		$max = 10,000$	
		Gap (%)	$t(s)$	Gap (%)	$t(s)$	Gap (%)	$t(s)$	Gap (%)	$t(s)$
A	459.9	0.00	0.083	0.00	0.180	0.00	2.020	0.00	1.677
B	666.6	0.00	0.297	0.00	0.367	0.00	3.810	0.00	3.615
C	604.8	0.00	6.781	3.47	0.775	2.08	7.053	1.88	7.761
D	804.6	0.00	11.534	6.04	1.356	3.50	11.431	2.76	13.942
E	730.2	0.00	17.325	15.98	2.349	12.08	18.922	11.75	19.744
F	863.7	0.00	24.527	17.37	3.496	12.19	26.594	11.98	28.146
G	753.3	0.04	31.552	32.91	5.246	26.53	44.204	25.14	45.608
H	890.1	0.03	45.017	33.38	7.083	27.61	57.517	25.69	73.439
I	812.1	0.07	77.217	44.18	10.844	39.27	90.182	37.42	110.032
Avg.:	**731.6**	**0.018**	**23.815**	**18.830**	**3.522**	**15.167**	**29.081**	**14.347**	**33.774**

Table 8
Comparison between Gaussian and Uniform initialization strategies.

Group	f_{LTM}	Gaussian		Uniform	
		Gap (%)	$t(s)$	*Gap* (%)	$t(s)$
A	459.9	0.00	0.083	0.00	0.093
B	666.6	0.00	0.297	0.00	0.326
C	604.8	0.00	6.781	1.83	12.987
D	804.6	0.00	11.534	0.74	30.740
E	730.2	0.00	17.325	1.81	66.191
F	863.7	0.00	24.527	1.94	117.476
G	753.3	0.04	31.552	19.16	187.054
H	890.1	0.03	45.017	20.80	198.258
I	812.1	0.07	77.217	33.79	104.285
Avg.:	**731.6**	**0.018**	**23.815**	**9.905**	**79.712**

As described in Section 5.1, the parameter setting of the EDA with uniform distribution with the values shown in Table 3 is performed by means of the Friedman test at a significance level equals to 0.05. The test indicates that there are statistically significant differences between the combinations of parameter values. The combinations with the best performance are those with $N = 2000$, $\alpha = \{20, 30\}$, $\beta = \{20, 30, 40\}$, $\gamma \in \{N/2, N\}$, $\lambda = 20$ and $p_{min} = \frac{1}{|Q| \cdot |\Omega|}$. In this case, the combination with $N = 2000$, $\alpha = \beta = 30\%$, $\gamma = N/2$, $\lambda = 20$ and $p_{min} = \frac{1}{|Q| \cdot |\Omega|}$ was selected because, compared with the remaining combinations, it provides a good ratio between objective function values and computational times.

Table 8 reports the comparison between both algorithms with respect to the LTM method. The first column (*Group*) provides the instance group to solve. Column f_{LTM} shows average objective function values obtained by the LTM method. Next columns (*Gaussian* and *Uniform*) show the results achieved by means of the EDA/LS when the proposed initialization scheme and the uniform distribution are considered, respectively. In both cases, average gaps (*Gap*) and average computational times (t), measured in seconds, are presented for each group of instances.

As can be seen in Table 8, the results obtained by means of this test demonstrate the suitability of the proposed initialization scheme to solve the QCSP. For each problem instance group, the quality of the achieved solutions in the first case is remarkably superior than in the uniform case. This situation is specially pronounced as the size of instances increases. That is, in the smallest instances (groups A and B) there is not any difference regarding the quality of the achieved solutions. The justification can be found in the reduced size of the search space, which allows to find the optimal solutions in a simple manner. However, the quality of the solutions found by means of the algorithm with a uniform distribution in large size instances is very poor. In these cases, the gap with respect to the results delivered by the LTM method is up to 33% on average, what makes it totally ineffective for this type of instances.

On the other hand, a detailed analysis need to be done according to the computational times of the experiment. As shown in the results, the initialization method noticeably influences the execution time of the algorithm. In this sense, there is almost a 56 s gap in the average computational times and a maximum gap larger than 155 s in instances from group G. Clearly, the convergence of the algorithm is affected by the selected initialization scheme. Initial probabilities of the model with uniform initialization have a poor performance with respect to the proposed scheme in order to sample high-quality solutions. Consequently, the algorithm needs to perform a large number of generations to reach promising regions.

5.6. Evaluation relaxation

Incorporating EETs into the structure of an optimization method can produce prominent changes on its performance. In this case, the assessment of the evaluation relaxation impact has been carried out considering 9 instances selected at random from the benchmark suite (one instance from each group). For each instance, 10^6 feasible schedules have been randomly generated and they have been evaluated by means of the accurate scheme proposed by Sammarra et al. [32] and the approximate evaluation scheme presented in Section 4.6. The relation of the computation effort required by both schemes in the evaluation of the related schedules was computed (Sastry et al. [33]). That is, $\eta = (t_{app} - t_{Sam})/(t_{Sam}) \times 100$, where t_{Sam} and t_{app} are the computational times required by the accurate and the approximate evaluation scheme, respectively. The results showed that $\eta < -95\,\%$ in all cases. Therefore, its use in the resolution of the QCSP is highly appropriate.

6. Conclusions

Exploring unidirectional schedules is an interesting approach to solve the QCSP since is the most widespread way to manage quay cranes in container terminals and allows to reach very high-quality solutions into a reduced search space. EDAs have demonstrated to be quite efficient optimization techniques to solve similar problems. This research proposes a hybrid EDA with local search (EDA/LS) aimed at detecting promising regions and perform intensification strategies, respectively. The features of the QCSP allow to include specific knowledge into the initialization step in order to speed up the search process. Additionally, a restarting strategy based on perturbing an adequate problem variable subset helps to prevent premature convergence and stagnation of the search. Lastly, an evaluation relaxation scheme is incorporated into the algorithm structure in order to reduce its computational burden.

Computational results show that the proposed scheme is highly competitive in all proposed scenarios with respect to already published algorithms. It allows to obtain optimal unidirectional solutions for small instances and high-quality solutions for the large ones. In the largest instances, the presented algorithm achieves the best quality solution in one instance. Moreover, the computational effort is very small and the robustness is high in each case, showing the convenience of applying it to real scenarios.

On the other hand, different practical constraints as container vessel stability, quay crane availability and interaction between quay cranes and other equipment of the container terminal should be considered in further researches. Finally, berthing and allocating of quay cranes are related problem at container terminal so their integration with QCSP is a promising topic for future study.

Acknowledgements

This work has been partially funded by the European Regional Development Fund, the Spanish Government (projects TIN2008-06872-C04-01, TIN2009-13363 and TIN2012-32608), and the Canary Government (project PI2007/019). Christopher Expósito Izquierdo thanks CajaCanarias the financial support he receives through his post-graduate grant.

References

[1] C.W. Ahn, J. An, J.-C. Yoo, Estimation of particle swarm distribution algorithms: combining the benefits of pso and edas, Information Sciences 192 (0) (2012) 109–119.
[2] C. Bierwirth, F. Meisel, A fast heuristic for quay crane scheduling with interference constraints, Journal of Scheduling 12 (2009) 345–360.
[3] M. Birattari, Tuning Metaheuristics: A Machine Learning Perspective, 1st ed., Springer Publishing Company, Incorporated, 2005 (2nd printing ed., 2009).
[4] M. Birattari, Z. Yuan, P. Balaprakash, T. Stützle, F-race and iterated f-race: an overview, in: T. Bartz-Beielstein, M. Chiarandini, L. Paquete, M. Preuss (Eds.), Experimental Methods for the Analysis of Optimization Algorithms, Springer, Berlin/Heidelberg, 2010, pp. 311–336.

[5] E.P. George, J. Box, S. Hunter, W.G. Hunter, Statistics for Experimenters: Design, Innovation, and Discovery, 2nd ed., Wiley-Interscience, 2005.
[6] S.-L. Chao, Y.-J. Lin, Evaluating advanced quay cranes in container terminals, Transportation Research Part E: Logistics and Transportation Review 47 (4) (2011) 432–445.
[7] J.H. Chen, D.-H. Lee, J. Xin Cao, Heuristics for quay crane scheduling at indented berth, Transportation Research Part E: Logistics and Transportation Review 47 (6) (2011) 1005–1020.
[8] S.-H. Chen, M.-C. Chen, Addressing the advantages of using ensemble probabilistic models in estimation of distribution algorithms for scheduling problems, International Journal of Production Economics 141 (1) (2013) 24–33.
[9] S.H. Chung, K.L. Choy, A modified genetic algorithm for quay crane scheduling operations, Expert Systems with Applications 39 (4) (2012) 4213–4221.
[10] W.W. Daniel, Applied Nonparametric Statistics, PWS-Kent Publishing Company, Boston, 1990.
[11] C. Fleurent, F. Glover, Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory, INFORMS Journal on Computing 11 (1999) 198–204.
[12] I. García del Amo, D. Pelta, Srcs: a technique for comparing multiple algorithms under several factors in dynamic optimization problems, in: E. Alba, A. Nakib, P. Siarry (Eds.), Metaheuristics for Dynamic Optimization, volume 433 of Studies in Computational Intelligence, Springer, Berlin/Heidelberg, 2013, pp. 61–77.
[13] M. Hauschild, M. Pelikan, An introduction and survey of estimation of distribution algorithms, Swarm and Evolutionary Computation 1 (3) (2011) 111–128.
[14] M.W. Hauschild, M. Pelikan, K. Sastry, D.E. Goldberg, Using previous models to bias structural learning in the hierarchical boa, in: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO'08, ACM, 2008, pp. 415–422.
[15] Y. Hong, S. Kwong, Y. Chang, Q. Ren, Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm, Pattern Recognition 41 (9) (2008) 2742–2756.
[16] C. Expósito Izquierdo, J. Luis González Velarde, B. Melián-Batista, J. Marcos Moreno-Vega, Estimation of distribution algorithm for the quay crane scheduling problem, in: D. Alejandro Pelta, N. Krasnogor, D. Dumitrescu, C. Camelia Chira, R. Lung (Eds.), Nature Inspired Cooperative Strategies for Optimization (NICSO 2011), Volume 387 of Studies in Computational Intelligence, Springer, Berlin/Heidelberg, 2012, pp. 183–194.
[17] C. Expósito Izquierdo, B. Melián-Batista, J. Marcos Moreno-Vega, An estimation of distribution algorithm for solving the quay crane scheduling problem with availability constraints, in: M. Grana, C. Toro, J. Posada, R.J. Howlett, L.C. Jain (Eds.), KES, Volume 243 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2012, pp. 10–19.
[18] C. Expósito Izquierdo, B. Melián-Batista, J. Moreno-Vega Marcos, Pre-marshalling problem: heuristic solution method and instances generator, Expert Systems with Applications 39 (9) (2012) 8337–8349.
[19] B. Jarboui, M. Eddaly, P. Siarry, An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems, Computers & Operations Research 36 (9) (2009) 2638–2646.
[20] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Computing 9 (2005) 3–12.
[21] K.H. Kim, Y.-M. Park, A crane scheduling method for port container terminals, European Journal of Operational Research 156 (3) (2004) 752–768.
[22] P. Legato, R. Trunfio, F. Meisel, Modeling and solving rich quay crane scheduling problems, Computers & Operations Research 39 (9) (2012) 2063–2078.
[23] H. Li, Y. Hong, S. Kwong, Subspace estimation of distribution algorithms: to perturb part of all variables in estimation of distribution algorithms, Applied Soft Computing 11 (3) (2011) 2974–2989.
[24] L. Wang, C. Fang, An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem, Computers & Operations Research 39 (2) (2012) 449–460, http://dx.doi.org/10.1016/j.cor.2011.05.008, issn: 0305-0548, http://www.sciencedirect.com/science/article/pii/S0305054811001328
[25] F. Meisel, The quay crane scheduling problem with time windows, Naval Research Logistics (NRL) 58 (7) (2011) 619–636.
[26] F. Meisel, C. Bierwirth, A unified approach for the evaluation of quay crane scheduling models and algorithms, Computers & Operations Research 38 (3) (2011) 683–693.
[27] L. Moccia, J.-F. Cordeau, M. Gaudioso, G. Laporte, A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal, Naval Research Logistics 53 (1) (2006) 45–59.
[28] M. Flavia Monaco, M. Sammarra, Quay crane scheduling with time windows, one-way and spatial constraints, International Journal of Shipping and Transport Logistics 3 (4) (2011) 454–474.
[29] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions i. binary parameters, in: H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature – PPSN IV, volume 1141 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 1996, pp. 178–187.
[30] E. Enda Ridge, D. Kudenko, Tuning an algorithm using design of experiments, in: T. Bartz-Beielstein, M. Chiarandini, L. Paquete, M. Preuss (Eds.), Experimental Methods for the Analysis of Optimization Algorithms, Springer, Berlin/Heidelberg, 2010, pp. 265–286.
[31] B. Saavedra-Moreno, S. Salcedo-Sanz, A. Paniagua-Tineo, L. Prieto, A. Portilla-Figueras, Seeding evolutionary algorithms with heuristics for optimal wind turbines positioning in wind farms, Renewable Energy 36 (11) (2011) 2838–2844.
[32] M. Sammarra, J.-F. Cordeau, G. Laporte, M. Monaco, A Tabu search heuristic for the quay crane scheduling problem, Journal of Scheduling 10 (2007) 327–336.
[33] K. Sastry, M. Pelikan, D. Goldberg, Efficiency enhancement of estimation of distribution algorithms, in: M. Pelikan, K. Sastry, E. Cantú-Paz (Eds.), Scalable Optimization via Probabilistic Modeling, Volume 33 of Studies in Computational Intelligence, Springer, Berlin/Heidelberg, 2006, pp. 161–185.
[34] R. Shah, P. Reed, Comparative analysis of multiobjective evolutionary algorithms for random and correlated instances of multiobjective d-dimensional knapsack problems, European Journal of Operational Research 211 (3) (2011) 466–479.
[35] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 4th ed., CRC Press, 2007.
[36] R. Stahlbock, S. Voß, Operations research at container terminals: a literature update, OR Spectrum 30 (2008) 1–52.
[37] D. Steenken, S. Voß, R. Stahlbock, Container terminal operation and operations research – a classification and literature review, OR Spectrum 26 (2004) 3–49.
[38] E.-G. Talbi, Metaheuristics: From Design to Implementation, John Wiley & Sons, 2009.
[39] I.F.A. Vis, R. de Koster, Transshipment of containers at a container terminal: an overview, European Journal of Operational Research 147 (1) (2003) 1–16.
[40] J. Xu, S.Y. Chiu, F. Glover, Fine-tuning a Tabu search algorithm with statistical tests, International Transactions in Operational Research 5 (3) (1998) 233–244.
[41] B. Yuan, M. Orlowska, S. Sadiq, Extending a class of continuous estimation of distribution algorithms to dynamic problems, Optimization Letters 2 (2008) 433–443.
[42] Q. Zhang, J. Sun, E. Tsang, Combinations of estimation of distribution algorithms and other techniques, International Journal of Automation and Computing 4 (2007) 273–280.
[43] Q. Zhang, J.g Sun, E. Tsang, J. Ford, Hybrid estimation of distribution algorithm for global optimisation, Engineering Computations 21 (1) (2003) 91–107.
[44] Q. Zhang, J. Sun, E. Tsang, J. Ford, Estimation of distribution algorithm with 2-opt local search for the quadratic assignment problem., in: J. Lozano, P. Larranaga, I. Inza, E. Bengoetxea (Eds.), Towards a New Evolutionary Computation, Volume 192 of Studies in Fuzziness and Soft Computing, Springer, Berlin/Heidelberg, 2006, pp. 281–292.