Ann Math Artif Intell (2016) 76:173–189
DOI 10.1007/s10472-014-9444-4

POPMUSIC as a matheuristic for the berth allocation problem

Eduardo Lalla-Ruiz · Stefan Voß

Published online: 31 December 2014

Abstract The Berth Allocation Problem aims at assigning and scheduling incoming vessels to berthing positions along the quay of a container terminal. This problem is a well-known optimization problem within maritime shipping. In order to address it, we propose two POPMUSIC (Partial Optimization Metaheuristic Under Special Intensification Conditions) approaches that incorporate an existing mathematical programming formulation. POPMUSIC is an efficient metaheuristic that may serve as blueprint for matheuristics approaches once hybridized with mathematical programming. In this regard, the use of exact methods for solving the sub-problems defined in the POPMUSIC template highlight an interoperation between metaheuristics and mathematical programming techniques, which provide a new type of approach for this problem. The computational experiments reveal excellent results.

Keywords Berth allocation problem · POPMUSIC · Matheuristic

Mathematics Subject Classification (2010) 90B06 · 90B80 · 90C90 · 68T20 · 90C59

1 Introduction

The increasing demand of maritime transport and the great competition among port terminals enforces terminal managers to efficiently exploit all resources in order to improve the competitiveness of the terminals. On the other hand, shipping companies often require

E. Lalla-Ruiz
Department of Computer and Systems Engineering, University of La Laguna, La Laguna, Spain
e-mail: elalla@ull.es

S. Voß (✉)
Institute of Information Systems, University of Hamburg, Hamburg, Germany
e-mail: stefan.voss@uni-hamburg.de

S. Voß
Escuela de Ingeniería Industrial, Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile

on-time and suitable services. In this context, according to [17], 93,6 % of the delayed vessel schedules are attributable to port access and terminal operations. These operations, referred to as seaside operations ([16], [23]), are strongly affected by the use of berths. Therefore, it becomes imperative for container terminals to suitably schedule their use to increase or maintain their market share ([27]).

The aforementioned discussion leads to the definition of the Berth Allocation Problem (BAP). Its goal is to determine the berthing position and berthing time for each container vessel arriving to the terminal over a given planning horizon. This problem has been extensively studied in the literature. While our practical intent on this goes back for about two decades [8], quite a bit of academic research has been done recently [2]. In this regard, due to the large variety of maritime terminal layouts, research has produced a multitude of variants for this problem.

(i) Depending on how the quay is modelled, the BAP can be referred to as discrete (the quay is divided into segments called berths) or continuous (the quay is not divided, thus the vessels can berth at any position along the quay). There are also options to discretize the continuous quay to have intermediate options from classical berths up to almost continuous ones [8]. Moreover, in some related works by, e.g., Cordeau et al. [6], Umang et al. [26], there is also a hybrid approach of the quay (it is divided into a set of berths and a vessel can occupy more than one berth at a time or share its assigned berth with other container vessels).

(ii) Depending on the arrival time, the BAP can be classified into *static*, that is, the vessels are already in the port when the berths become available; or *dynamic*, that is, the vessels arrive during the planning horizon. Hence, the dynamic variant considers that the vessels arrive at a certain time within the planning horizon.

(iii) Depending on the planning level, the BAP can be addressed at different levels. Namely, operational, tactical or strategic level. At the operational level, the decisions cover daily operations, while at the tactical level decisions from one week to several months are addressed. Finally, the strategic level covers a time horizon from one to several years.

For detailed descriptions of the BAP, the reader is referred to the works of Bierwirth and Meisel [2] and Christiansen et al. [5]. The first of these two papers also provides a comprehensive literature review on berth allocation and related problems so that the reader may consult this survey for additional references. To this end we also see a wealth of extensions like the integrative treatment of berth allocation with quay crane allocation (crane split) or the scheduling of yard resources (like automated guided vehicles, gantry cranes, etc.). A few recent papers include, e.g., [9, 11, 14, 19, 21, 28, 29].

In this paper, we study the Dynamic Berth Allocation Problem (DBAP) proposed by Cordeau et al. [6], that considers the berth's and vessel's time windows as well as heterogeneous vessel service times depending on the assigned berth. According to the classification shown below, in the DBAP the dynamic discrete version of the BAP is addressed at an operational level. This problem is $\mathcal{NP}$-hard since it can be modeled as a Multi-Depot Vehicle Routing Problem with Time-Windows (MDVRPTW). Due to its difficulty, several approximate and exact solution approaches have been proposed. Recently, de Oliveira et al. [18] investigated a Clustering Search with Simulated Annealing and Ting et al. [25] applied Particle Swarm Optimization for solving the DBAP. Although they are able to provide the optimal solution values for the biggest instances proposed by Cordeau et al. [6], they are not able to ensure that behaviour. Moreover, the mathematical reformulation of this problem as a Generalized Set-Partitioning Problem allows to solve those instances within reasonable

computational time (see Buhrkal et al. [3]). However, Lalla-Ruiz et al. [12] pointed out that the GSPP model implemented in CPLEX runs out of memory for problem instances where other characteristics are taken into account.

Large optimization problems, like the DBAP, usually require a lot of computational effort. A natural way to solve these problems is by decomposing them into independent sub-problems that are solved with an appropriate procedure (Resende and Pardalos [20]). Nevertheless, such strategies can require a non-trivial way to address the decomposition of the problem, which may also lead to solutions of moderate quality since the sub-problems might have been created in a somewhat arbitrary fashion. Finding an appropriate way to decompose a problem 'a priori' can be difficult, despite the fact that the decomposition can be problem-dependent as it is designed according to a specific problem. Due to this, we will need to design a specific decomposition for each problem. For addressing these concerns, Taillard and Voß [24] propose the POPMUSIC framework. Its basic idea is to locally optimize sub-parts of a solution, 'a posteriori,' once a solution to the problem is available. These local optimizations are repeated until a local optimum is found. POPMUSIC may be viewed as a local search working with a special, large neighbourhood.

The goals of this work are, on one hand, to assess the behaviour of POPMUSIC using an exact method for solving the sub-problems. In this sense, the method becomes similar to the corridor method of [22]. On the other hand, we seek to evaluate the effectiveness of the POPMUSIC template by comparing its computational results with those given by the exact resolution of the mathematical model presented by Christensen and Holst [4] and the results obtained by the best algorithms in the related literature for this problem. In this regard, as it will be discussed below, the results provided by POPMUSIC corroborate that it is able to solve large scale instances from the literature to optimality and provide new best values for instances where the mathematical model implemented in a general-purpose solver runs out of memory.

The remainder of this paper is organized as follows. The berth allocation problem is depicted in more detail in Section 2. In Section 3, the POPMUSIC approach for addressing the dynamic berth allocation problem is described. The computational experience carried out and a comparison summary are presented in Section 4. Finally, some conclusions and several lines for further research are drawn in Section 5.

2 The dynamic berth allocation problem

In this paper, we study the application of POPMUSIC for solving the discrete Dynamic Berth Allocation Problem (DBAP) proposed by Cordeau et al. [6]. In this problem we are given a set of incoming container vessels, N, and a set of berths, M. Each container vessel, $i \in N$, must be assigned to an empty berth, $k \in M$, within its time window, $[t_i, t_i']$, and the assigned berth time window, $[s_k, e_k]$. Each berth can handle at most one vessel at a time. For each container vessel, $i \in N$, its handling time, ρ_{ik}, depends on the berth $k \in M$ where it is assigned to. That is, the handling time of a given vessel differs from one berth to another. Moreover, some vessels may have forbidden berths in order to include water-depth or maintenance constraints. Finally, each vessel $i \in N$ has a given service priority, denoted as v_i, according to its contractual agreement with the terminal. It should be noted that the higher this value, the higher the priority of the vessel. A comprehensive description of the DBAP is provided by Cordeau et al. [6], Imai et al. [10], and Lalla-Ruiz et al. [12].

The objective of the DBAP is to minimize the total (weighted) service time of the incoming container vessels. In this context, the service time of a container vessel is defined as the

time elapsed between its arrival to the port and the completion of its transshipment operations. A mathematical formulation of the DBAP is provided by Cordeau et al. [6]. While this formulation is based on a generalization of the vehicle routing problem, below we discuss one based on a generalized set partitioning problem.

2.1 An example

Figure 1 illustrates an example of a solution for the DBAP. In the figure, a schedule and an assignment plan are shown for six vessels within three berths. The rectangles indicate the vessels and inside each rectangle we display the service priority of each vessel (v_i), used for establishing vessel priorities. The time windows of the vessels are represented by the lines at the bottom of the figure. In this case, for example, vessel 1 arrives at time step 4 and it should be served until time step 14. Moreover, the time window of each berth is limited by the non-hatched areas. Table 1 reports the different handling times (ρ_{ik}) for each vessel (i) depending on the assigned berth (k). For example, if vessel 1 is assigned to berth 1, its handling time would be equal to 6, which is shorter than the handling time of 8 that it would have at berth 2.

As can be seen in the example, vessels 5 and 6 would have to wait for berthing in their respective assigned berths. In this regard, since their service priority value is $v_5 = v_6 = 1$, their waiting will have less impact on the objective function value than delaying other vessels, like vessels 3 and 4, for which the service priorities are $v_3 = 6$ and $v_4 = 4$, respectively. That is, if their berthing time is delayed, the waiting time step of each vessel is multiplied by 6 and 4, respectively.

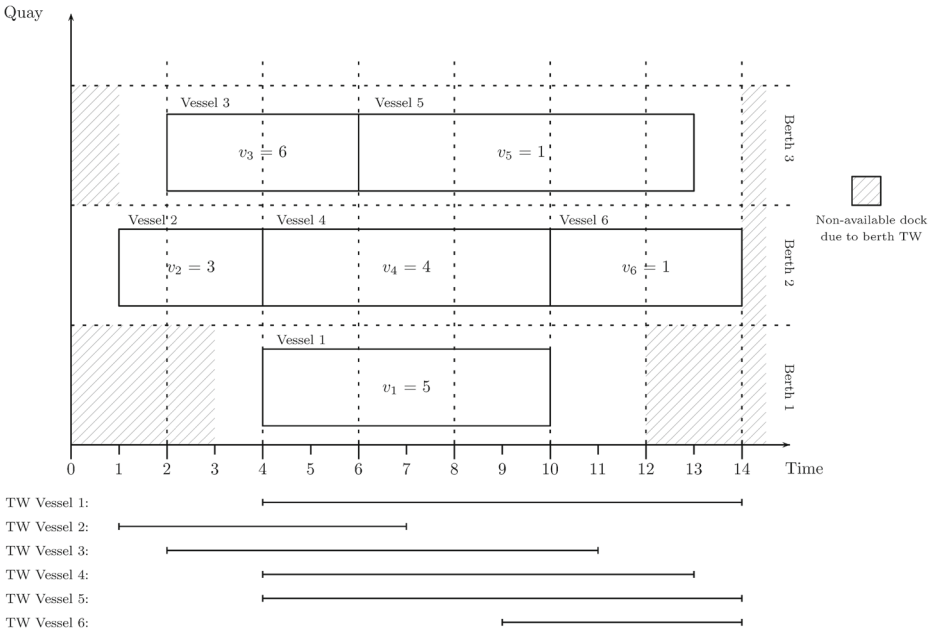

Fig. 1 Example of a solution for the DBAP with six vessels and three berths

Table 1 Vessel handling times depending on the berth

	Berth 1	Berth 2	Berth 3
Vessel 1	6	8	5
Vessel 2	2	3	4
Vessel 3	5	5	4
Vessel 4	4	6	5
Vessel 5	5	8	7
Vessel 6	4	4	5

As indicated above, the objective value of a solution for the DBAP is the total (weighted) service time of the incoming container vessels. In this example, the weighted service times of the six vessels are calculated as follows.

- **Vessels 1-4**. Since the vessels are served as soon as they arrive to the port, their weighted service time is equal to their handling times times their priority ($\rho_{ik} \cdot v_i$). The contribution of each of these vessels to the objective function value of the solution is as follows.
  - Vessel 1: $\rho_{11}(6) \cdot v_1(5) = 30$
  - Vessel 2: $\rho_{22}(3) \cdot v_2(3) = 9$
  - Vessel 3: $\rho_{33}(4) \cdot v_3(6) = 24$
  - Vessel 4: $\rho_{42}(6) \cdot v_4(4) = 24$
- **Vessels 5-6** These vessels have to wait after their arrival to the port in order to be serviced. Their weighted service time is equal to the time required to be serviced from their arrival to their departure (waiting time plus handling time) times their priority. In this case, the contribution of each of these vessels to the objective function value of the solution is:
  - Vessel 5: $9 \cdot v_5(1) = 9$
  - Vessel 6: $5 \cdot v_6(1) = 5$

Therefore, the objective function value of this solution is 101.

2.2 Generalized set-partitioning problem formulation

Christensen and Holst [4] propose a reformulation of the DBAP as a Generalized Set-Partitioning Problem (GSPP). Buhrkal et al. [3] use this formulation to perform a comparison of the different models for solving the DBAP and show that the GSPP is superior to all other models on the set of instances from Cordeau et al. [6]. In order to make this paper self-contained, we include the mathematical formulation of the GSPP.

The GSPP formulation is as follows. A column represents a feasible assignment of a vessel to a berth at a time. The set of columns is denoted by Ω. Two matrices A and B are defined, both containing $|\Omega|$ columns. Matrix $A = (A_{i\omega})$ contains a row for each vessel, and $A_{i\omega} = 1$, if and only if column ω represents an assignment of vessel $i \in N$. Each column of A contains exactly one non-zero element. Matrix $B = (B_{p\omega})$ contains a row per (berth,time) position.

The rows of B are indexed by the set $P = \{1, 2, ..., K\}$ with $K = \sum_{k \in M}(e_k - s_k)$. The entry $B_{p\omega}$ is equal to 1, if and only if, position $p \in P$ is contained in the assignment that column ω represents. The cost c_ω of any column $\omega \in \Omega$ is the service time of the respective

position assignment and can be multiplied by the priority factor v_i if necessary. A binary variable x_ω is equal to 1, if column ω is used in the solution, and 0 otherwise. With these definitions the GSPP formulation of the BAP presented in [3] is stated as follows.

$$\text{minimize} \sum_{w \in \Omega} c_w x_w \tag{1}$$

subject to

$$\sum_{w \in \Omega} A_{iw} x_w = 1 \quad \forall i \in N, \tag{2}$$

$$\sum_{w \in \Omega} B_{pw} x_w \leq 1 \quad \forall p \in P, \tag{3}$$

$$x_w \in \{0, 1\} \quad \forall w \in \Omega \tag{4}$$

The objective function (1) minimizes the service time of the vessels. The set of constraints (2) ensures that all vessels are served. Finally, the constraints (3) guarantee that at a time interval, in a berth, only one vessel can be served.

3 Partial optimization metaheuristic under special intensification conditions (POPMUSIC)

3.1 The general framework

POPMUSIC was firstly proposed by Taillard and Voß [24] as a framework for tackling large problem instances. Its basic idea is to split a solution of the problem at hand, S, into h parts $part_1$, $part_2$, ..., $part_h$ and joining some of them to build a sub-problem, R. To form the sub-problem one of the h parts is selected, $part_{seed}$, termed as *seed-part*. Once that, a number, r, of the closest parts to $part_{seed}$ are aggregated to the latter to form the sub-problem R. In order to determine the closeness of the parts a distance measure among them is defined. Once a sub-problem is constructed, it is solved by using an approximate or exact solution approach. If parts and sub-problems are defined in an appropriate way, every improvement of a sub-problem corresponds to an improvement of the whole solution S. This process is repeated, e.g., until the solution does not contain a sub-problem that can be improved.

Algorithm 1: POPMUSIC framework

```
1  Generate an initial solution S
2  Decompose S in h parts, H = {part_1, ..., part_h}
3  Set O = ∅
4  while O ≠ {part_1, ..., part_h} do
5      Select a seed part part_seed ∉ O
6      Build a sub-problem R composed of the r parts of S which are the closest to
       part_seed
7      Optimize R by using an approximate or exact solution approach
8      if R has been improved then
9          Update solution S
10         O ← ∅
11     else
12         Include part_seed in O
13 return the improved solution S
```

In Algorithm 1 the POPMUSIC framework is depicted. Firstly, an initial solution is generated, S (line 1). The next step is to divide the solution into h parts (line 2). Then, a seed part, $part_{seed}$, is selected (line 5). A sub-problem, R, is constructed by considering its r nearest parts according to the relatedness function (line 6). In this regard, the unique parameter of this framework, r, is used for delimiting the size of the sub-problems. The sub-problem R is then solved by an approximate or exact procedure (line 7). In this framework, the set of parts O corresponds precisely to seed parts that have been used to define sub-problems that have been unsuccessfully optimized. Once O contains all the parts of the complete solution (line 4), the process stops as all sub-problems have been examined without success.

3.2 POPMUSIC approaches for the DBAP

In the context of the DBAP, the POPMUSIC implementation considers a solution S as a sequence composed by the vessel identifiers, where each berth is delimited by a 0. The service order of each vessel is determined by its position in the sequence. The solution structure for the example of Fig. 1 for three berths and six vessels is as follows: $S = \{1, 0, 2, 4, 6, 0, 3, 5\}$.

The configuration of the parts for the DBAP is defined by means of the berths. That is, the number of parts, h, is equal to $|M|$. Hence, for example, for the solution structure reported above, $S = \{1, 0, 2, 4, 6, 0, 3, 5\}$, the number of parts will be equal to 3 since there are three berths. The structure of each part is determined by the vessels within them. That is, $part_1 = \{1\}$, $part_2 = \{2, 4, 6\}$ and $part_3 = \{3, 5\}$. As can be seen from the example, the size of the parts depends on the number of vessels assigned to the berth represented by the part.

Since the parts are determined by the vessels assigned to each berth, we define the distance of a part to the others by the forward distance among its identifier, id, and its increase in r units, $id + r$. Hence, a sub-problem R is composed of $part_{id}$, $part_{id+1}$, ..., $part_{id+r}$. In case $id + r$ is higher than the number of parts, then the neighbour parts are considered starting from 1. Considering the above-mentioned example, $part_1$, which is related to the berth 1, has as a 'neighbour part', $part_2$ for $r = 1$, and has as 'neighbour parts' $part_2$ and $part_3$ when $r = 2$. In the case of $part_3$, its neighbour solution for $r = 1$ is $part_1$.

Lalla-Ruiz et al. [12] point out that the GSPP mathematical formulation implemented in CPLEX is not able to provide any solution for the majority of medium- and large-sized instances proposed in that work. Namely, as the time horizon, the number of vessels/berths and the time windows are increased, CPLEX runs out of memory as it requires a large amount of memory for providing a feasible solution using a standard computer. In this context, the use of the POPMUSIC framework offers the advantage of dividing these large-sized problem instances into sub-problems that can be treated by CPLEX and provide feasible solutions. Moreover, POPMUSIC allows to narrow those memory-consuming problem constraints in order to enhance the performance of the GSPP mathematical formulation implemented in CPLEX. The use of exact methods for solving the sub-problems embedded in the POPMUSIC framework point out an interoperation between metaheuristics and mathematical programming techniques, which in the related literature is denoted as matheuristic [15].

As indicated above, the POPMUSIC approach solves each sub-problem using its GSPP mathematical formulation implemented in CPLEX. Hence, for each sub-problem, as contemplated in the GSPP formulation (see Section 2.2), we generate the feasible columns for the set of vessels and berths specified in the sub-problem. For example, continuing the

previous example, let a sub-problem R consist of $part_1 = \{1\}$ and $part_2 = \{2, 4, 6\}$. Then, through the GSPP formulation, we will solve the sub-problem consisting of the set of vessels $N' = \{1, 2, 4, 6\}$ and berths $M' = \{1, 2\}$ with their detailed information (service times, time windows, arrival times, etc.) from the complete instance. Note, that the columns are built as indicated in Section 2.2 using only the information from the sub-problem.

Once the POPMUSIC process is over, all the solution parts are joined. The information obtained from them is used for determining a reduced problem instance that will be provided to CPLEX. This narrow problem allows CPLEX to solve this reduced problem to optimality without running out of memory. In doing so, we identify those characteristics provided by the POPMUSIC solution regarding those problem constraints that can be narrowed. These constraints are:

(i) Each vessel $i \in N$ can only be berthed at berth $k \in M$ until k becomes unavailable at time step e_k.
(ii) Each vessel $i \in N$ has to be serviced until its departure time t'_i.

The way the constraints are narrowed is by tightening their related data from the problem instance as explained below.

- We identify the maximum berth finalization time, max_B from the POPMUSIC solution when joining the solution parts. This value is equal to the last departure time within all vessels. With that value, for each e_k, $k \in M$ from the problem instance data, if $e_k > max_B + (max_B)/h$, it will be reduced to $max_B + (max_B)/h$.
- For each t'_i, $i \in N$ from the problem instance data, if $t'_i > max_B + (max_B)/h$, it will be reduced to $max_B + (max_B)/h$.

The addition of $(max_B)/h$ is used as a relaxation looseness in order to consider the relation that berths (parts) may have for allocating vessels within them when solving the complete problem. As will be discussed in Section 4, the problem instances proposed in the related literature for this problem by Cordeau et al. [6] and Lalla-Ruiz et al. [12] do not take into account any priority of the vessels, i.e., $v_i = 1, i \in N$.

For solving the DBAP we propose two different POPMUSIC approaches, *POPMUSIC* and *POPMUSIC-G*. In Algorithm 2, the POPMUSIC framework adapted for the DBAP is shown. The initial solution S is randomly generated (line 1) by applying a random-greedy method (R-G) proposed by Cordeau et al. [6], given a random vessel permutation. Vessels are assigned one at a time to the best possible berth following the sequence order. Then, the solution is divided into a set H of parts, depending on the number of berths (line 2). The set of improved parts, O, is emptied (line 3). A seed part, s_{seed}, is selected at random from the set of parts, H (line 5). Once a solution part is selected, the sub-problem R is established by joining the s_{seed} and its r neighbour parts (line 6). The GSPP mathematical formulation of the sub-problem R is solved by CPLEX (line 7). In case, R has been improved, then the solution S is updated (line 9) and the set O is emptied (line 10). Otherwise, s_{seed} is included in O (line 12). This procedure (lines $5 - 12$) is repeated until O is fulfilled with all the h parts (line 4); then the improved solution is provided (line 13).

The POPMUSIC-G differs from the original in the way the set of parts O is fulfilled when there is an improvement. That is, in case a sub-problem R is improved, all its composing parts (s_{seed} and its neighbour parts) are included in O, instead of emptying O like in the standard approach.

Algorithm 2: POPMUSIC Approach for the DBAP

```
1  Generate an initial solution S at random using R-G
2  Decompose S in M parts according to the number of berths, H = {part_1, ..., part_M}
3  Set O = ∅
4  while O ≠ {part_1, ..., part_M} do
5      Select a seed part s_seed ∈ H at random and s_seed ∉ O
6      Build a sub-problem R composed of the r parts of S which are the closest to s_seed
7      Optimize R through solving its GSPP mathematical formulation using a
       general-purpose solver
8      if R has been improved then
9          Update solution S
10         O ← ∅
11     else
12         Include s_seed in O
13 return the improved solution S
```

As indicated in [1] and [24] the time complexity of this template is high since it depends on set O, which might not be reduced at each iteration and can be emptied. Nevertheless, in [1] the authors indicate that empirically the complexity grows quasi-linearly with the number of parts. In this regard, since POPMUSIC-G includes also the neighbouring parts into the set O, the set O is filled before and, therefore, its time complexity is lower than the traditional POPMUSIC approach. This feature is shown in Section 4, where the time performance of POPMUSIC-G is better than POPMUSIC in terms of computational time. Moreover, the complexity of POPMUSIC, as discussed in Section 4.1, depends on the generation of a decent initial solution since the better the solution the lesser local improvements are required. Finally, it should be noted that since POPMUSIC is a framework, its computational performance in terms of running time depends on the optimizing technique used within it.

4 Computational results

This section is devoted to present the computational experiments carried out for assessing the performance of the proposed methods. All the reported computational experiments were conducted on a computer equipped with an Intel 3.16 GHz and 4 GB of RAM.

The problem instances used for evaluating the proposed algorithm are the large-sized instances proposed by Cordeau et al. [6] with 60 vessels and 13 berths. Those instances were generated by taking into account a statistical analysis of the traffic and berth allocation data at the maritime container terminal of Gioia Tauro (Italy) [7]. Moreover, we also use a representative set of instances from the ones proposed by Lalla-Ruiz et al. [12] in order to study the behaviour of POPMUSIC for instances where the GSPP implemented in CPLEX using a standard computer runs out of memory. In doing so, we selected a representative group of them.

4.1 Parameter and robustness assessment

In this subsection the selection of the parameter r, the initialization method, and the robustness of our method are discussed. In doing so, on the one hand, for the initialization

method, we use the two most common ones proposed in the literature for this problem. Namely:

- *Random-greedy method (R-G)* proposed by Cordeau et al. [6]. Given a random vessel permutation, vessels are assigned one at a time to the best possible berth by means of the objective function value following the sequence order.
- *First-Come First-Served Greedy (FCFS-G)* proposed by Cordeau et al. [6] is based on the First-Come First-Served queue policy used at some container terminals for allocating vessels. The vessels are ordered according to their arrival times and, then, assigned one at a time to the best possible berth by means of the objective function value following the sequence order.

On the other hand, for assessing the robustness of our approaches we made a comparison over a reduced group of those instances tackled in this paper. In this regard, for each instance, we report the success rate ($\alpha(\%)$) based on the number of times the best value known for that instance is obtained over the total number of replications such that,

$$\alpha = \frac{number\ of\ best\ known\ solutions\ provided}{number\ of\ replications}(\%) \tag{5}$$

where 100 % indicates that in all the replications the best solution known for that instance has been provided.

Table 2 illustrates the results provided by POPMUSIC and POPMUSIC-G for a set of instances proposed in [6, 12]. We run both algorithms 10 times, for $r = 1$. The first column in the tables reports the problem instances used. For each approach (POPMUSIC and POPMUSIC-G) and for each starting method (R-G and FCFS), we report the average objective function value (Avg. Obj.), the average computational time (Avg. t(s.)) measured in seconds, and the success rate of the algorithm in terms of the best-known solutions provided for that instance (α (%)). In the table it can be verified that FCFS-G exhibits a better performance than R-G in terms of running time. On the other hand, POPMUSIC-G exhibits a similar temporal performance regardless the initialization method.

Moreover, we have verified that the $\alpha = 100$ % percentage is maintained for any of the r values considered in this work, that is, for $r = 1, 2, 3, 4$. Therefore, in Table 3, we only report the average computational times reported by POPMUSIC and POPMUSIC-G for each r value over 10 replications using the FCFS-G. As it can be seen, the bigger r the longer the running time. Concerning both methods, it can be checked that POPMUSIC-G exhibits a better temporal performance than POPMUSIC.

4.2 Comparison with literature approaches

In this subsection we report the comparison with different approaches proposed in the literature for this problem. At the light of the results provided in the previous subsection, in the following we report the computational results for $r = 1$ using FCFS-G as an initialization method for both, POPMUSIC and POPMUSIC-G.

For the instances proposed by Cordeau et al. [6] we provide a comparison among:

(a) The best exact solution approach (GSPP) proposed in the related literature by Christensen and Holst [4] implemented in CPLEX for this work.
(b) The best approximate solution approaches proposed in the literature for this problem.
    (i) Clustering-Search with Simulated-Annealing (CS-SA) from de Oliveira et al. [18].

Table 2 Computational results in terms of POPMUSIC and POPMUSIC-G for different starting procedures (R-G and FCFS)

	POPMUSIC						POPMUSIC-G					
	R-G			FCFS-G			R-G			FCFS-G		
	Avg. Obj.	Avg. t(s.)	α(%)	Avg. Obj.	Avg. t(s.)	α(%)	Avg. Obj.	Avg. t(s.)	α(%)	Avg. Obj.	Avg. t(s.)	α(%)
i01	1409	34.84	100	1409	10.22	100	1409	10.17	100	1409	10.51	100
i02	1261	32.29	100	1261	9.53	100	1261	9.31	100	1261	9.88	100
i03	1129	31.21	100	1129	8.78	100	1129	8.59	100	1129	8.86	100
i04	1302	26.15	100	1302	9.72	100	1302	9.25	100	1302	9.93	100
i05	1207	32.29	100	1207	12.54	100	1207	9.40	100	1207	9.39	100
$40 \times 5 - 03$	2880	132.87	100	2880	130.56	100	2301	82.76	100	2301	78.87	100
$40 \times 7 - 03$	2119	144.95	100	2119	103.16	100	2119	58.32	100	2119	48.86	100
$55 \times 5 - 06$	5595	297.91	100	5595	325.89	100	5595	122.95	100	5595	93.71	100
$55 \times 7 - 03$	3825	221.88	100	3825	185.74	100	3825	85.15	100	3825	106.71	100
$60 \times 5 - 03$	6780	322.51	100	6780	246.17	100	6780	103.96	100	6780	120.17	100
Average	2750.7	127.69	100	2750.7	104.23	100	2692.8	49.99	100	2692.8	49.69	100

The reported objective values and times are the average calculated from all the executions.

Table 3 Temporal performance of POPMUSIC and POPMUSIC-G in terms of average running time for different r values, namely, $r = 1, 2, 3, 4$

	POPMUSIC				POPMUSIC-G			
	$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 1$	$r = 2$	$r = 3$	$r = 4$
i01	10.22	17.93	33.44	67.25	10.51	13.45	18.47	22.77
i02	9.53	14.03	33.83	55.81	9.88	14.27	16.68	16.55
i03	8.78	13.43	39.94	71.15	8.86	14.05	19.62	20.17
i04	9.72	15.43	39.50	65.29	9.93	14.98	14.59	21.47
i05	12.54	17.85	41.23	73.08	9.39	13.37	17.19	17.01
$40 \times 5 - 03$	130.56	234.22	346.22	856.84	78.87	98.77	139.56	201.27
$40 \times 7 - 03$	103.16	393.44	435.63	411.93	48.86	61.63	74.63	90.17
$55 \times 5 - 06$	325.89	265.28	590.18	198.58	93.71	176.67	229.93	197.34
$55 \times 7 - 03$	185.74	429.93	708.92	858.44	106.71	119.67	142.73	240.81
$60 \times 5 - 03$	246.173	653.19	647.83	100.98	120.17	137.83	521.81	100.24
Average	104.23	205.47	291.67	275.94	49.69	66.47	119.52	92.78

The clustering-search method is an iterative approach decomposing the search space into clusters that can be searched by means of some metaheuristic like simulated annealing.

(ii) Particle Swarm Optimization (PSO) from Ting et al. [25].
The PSO aims at optimizing the problem by iteratively trying to improve a population of candidate solutions called particles and 'moving' these particles within the search space. The moves of each particle are based on their local best position as well as some guidance towards best known positions in the search space.

(iii) Tabu Search with Path Relinking, T^2S^*+PR, as proposed by Lalla-Ruiz et al. [12]. In tabu search information about the search history is used to guide a local search approach to overcome local optimality. Based on some sort of memory, certain moves may be forbidden. Within this approach path relinking provides a useful means of intensification and diversification where a path is constructed between different solutions with the aim to find improved ones.

(c) The matheuristic approaches, POPMUSIC and POPMUSIC-G.

For the instances proposed by Lalla-Ruiz et al. [12] we made a comparison among:

(a) The best exact solution approach (GSPP) implemented in CPLEX for this work.
(b) The best approximate solution approach algorithm, T^2S^*+PR, proposed in the related literature for those instances by Lalla-Ruiz et al. [12].
(c) The matheuristic approaches, POPMUSIC and POPMUSIC-G.

Table 4 illustrates the results obtained for the instances proposed by [6]. The first column corresponds to the problem instance. In the GSPP column we report the optimal value (Opt.) and the required computational time (*t(s.)*). In columns POPMUSIC and POPMUSIC-G, we report the average objective value obtained from all the executions (Avg. Obj.) and the average computational time (*t(s.)*). For the T^2S^*+PR, CS, and PSO, we report their best value (Obj.), relative error (Gap(%)) and running time (t(s.)).

Table 4 Results for the problem instances proposed by Cordeau et al. [6]

	GSPP		POPMUSIC		POPMUSIC-G		T^2S^*+PR			CS			PSO		
	Opt.	t(s.)	Avg. Obj.	t(s.)	Avg. Obj.	t(s.)	Obj.	Gap (%)	t(s.)	Obj.	Gap (%)	t(s.)	Obj.	Gap (%)	t(s.)
i01	1409	33.20	1409	10.05	1409	9.27	1409	0.00	8.70	1409	0.00	12.47	1409	0.00	11.11
i02	1261	29.18	1261	9.53	1261	8.67	1261	0.00	8.44	1261	0.00	12.59	1261	0.00	7.89
i03	1129	28.17	1129	8.78	1129	7.43	1129	0.00	9.20	1129	0.00	12.64	1129	0.00	7.48
i04	1302	29.20	1302	9.72	1302	7.89	1302	0.00	8.97	1302	0.00	12.59	1302	0.00	6.03
i05	1207	27.93	1207	12.54	1207	8.78	1207	0.00	8.92	1207	0.00	12.68	1207	0.00	5.84
i06	1261	29.75	1261	11.83	1261	11.71	1261	0.00	8.19	1261	0.00	12.56	1261	0.00	7.67
i07	1279	32.89	1279	9.82	1279	10.02	1279	0.00	8.88	1279	0.00	12.63	1279	0.00	7.50
i08	1299	30.19	1299	12.31	1299	12.66	1299	0.00	9.64	1299	0.00	12.57	1299	0.00	9.94
i09	1444	30.89	1444	10.74	1444	10.97	1444	0.00	8.94	1444	0.00	12.58	1444	0.00	4.25
i10	1213	29.14	1213	9.68	1213	9.73	1213	0.00	8.84	1213	0.00	12.61	1213	0.00	5.20
i11	1368	30.62	1368	11.45	1368	11.73	1368	0.00	10.28	1368	0.00	12.58	1368	0.00	10.52
i12	1325	28.93	1325	12.35	1325	11.76	1326	0.08	10.22	1325	0.00	12.61	1325	0.00	12.92
i13	1360	30.14	1360	10.90	1360	11.17	1360	0.00	8.19	1360	0.00	12.58	1360	0.00	11.97
i14	1233	26.17	1233	13.08	1233	10.35	1233	0.00	9.75	1233	0.00	12.56	1233	0.00	7.11
i15	1295	26.59	1295	13.85	1295	10.04	1295	0.00	8.50	1295	0.00	12.61	1295	0.00	8.30
i16	1364	24.65	1364	10.05	1364	10.24	1364	0.00	10.14	1364	0.00	12.67	1364	0.00	8.48
i17	1283	24.89	1283	10.41	1283	10.63	1283	0.00	8.17	1283	0.00	13.80	1283	0.00	5.66
i18	1345	23.17	1345	11.05	1345	11.31	1345	0.00	9.78	1345	0.00	14.46	1345	0.00	8.02
i19	1367	24.66	1367	11.40	1367	11.59	1367	0.00	10.00	1367	0.00	13.73	1367	0.00	11.42
i20	1328	30.77	1328	11.66	1328	11.96	1329	0.08	10.94	1328	0.00	12.82	1328	0.00	12.28
i21	1341	24.59	1341	18.00	1341	9.88	1341	0.00	9.56	1341	0.00	12.68	1341	0.00	7.11
i22	1326	23.76	1326	16.96	1326	9.52	1326	0.00	10.36	1326	0.00	12.62	1326	0.00	7.94
i23	1266	22.72	1266	10.67	1266	11.02	1266	0.00	8.73	1266	0.00	12.62	1266	0.00	7.25

Table 4 (continued)

	GSPP		POPMUSIC		POPMUSIC-G		T^2S^*+PR			CS			PSO		
	Opt.	t(s.)	Avg. Obj.	t(s.)	Avg. Obj.	t(s.)	Obj.	Gap (%)	t(s.)	Obj.	Gap (%)	t(s.)	Obj.	Gap (%)	t(s.)
i24	1260	21.06	1260	12.95	1260	9.13	1260	0.00	10.38	1260	0.00	12.64	1260	0.00	5.67
i25	1376	25.43	1376	19.04	1376	10.51	1376	0.00	10.20	1376	0.00	12.62	1376	0.00	7.13
i26	1318	23.98	1318	11.70	1318	13.18	1318	0.00	9.58	1318	0.00	12.62	1318	0.00	7.44
i27	1261	22.82	1261	10.49	1261	10.26	1261	0.00	9.09	1261	0.00	12.64	1261	0.00	6.16
i28	1359	28.01	1359	14.22	1359	10.14	1360	0.07	9.20	1359	0.00	12.71	1359	0.00	11.52
i29	1280	21.34	1280	10.47	1280	10.21	1281	0.08	9.19	1280	0.00	12.62	1280	0.00	8.11
i30	1344	29.84	1344	9.71	1344	9.93	1344	0.00	9.69	1344	0.00	12.58	1344	0.00	7.13
		27.16		11.85		10.39			9.36			12.76			8.17

We report POPMUSIC and POPMUSIC-G average objective values of all the executions.

Table 5 Results for a representative set of instances from the ones proposed by Lalla-Ruiz et al. [12]

	GSPP		POPMUSIC		POPMUSIC-G		T^2S^*+PR		
	Opt.	t(s.)	Avg. Obj.	Avg. t(s.)	Avg. Obj.	Avg. t(s.)	Obj.	Gap (%)	t(s.)
40x5-01	2301	41.51	2301	102.92	2301	32.57	2303	0.09	0.90
40x5-02	2829	59.89	2829	134.74	2829	53.67	2834	0.18	1.09
40x5-03	2880	99.20	2880	130.56	2880	78.87	2880	0.00	0.50
40x7-03	—	—	2119	103.16	2119	48.86	2119	0.00*	1.17
55x5-03	—	—	5499	273.23	5499	107.55	5499	0.00*	2.67
55x5-05	—	—	5478	311.42	8478	139.85	5478	0.00*	2.73
55x5-06	—	—	5595	325.89	5595	93.71	5595	0.00*	2.56
55x7-03	—	—	3825	185.74	3825	106.71	3833	0.21*	5.57
55x7-05	—	—	3797	227.19	3797	159.26	3801	0.11*	3.56
55x7-06	—	—	3783	173.69	3783	107.97	3789	0.16*	3.70
60x5-03	—	—	6780	246.17	6780	120.17	6780	0.00*	4.25
60x5-06	—	—	6616	349.45	6616	144.96	6616	0.00*	3.53

(*)indicates those gaps that are calculated considering the POPMUSIC results.

As can be seen in Table 4, both approaches, POPMUSIC and POPMUSIC-G provide the optimal solutions in all cases and executions. Note that the average value coincides with the optimal value, which means $\alpha = 100\%$ (see Section 4.1). This characteristic points out that the recognition of 'useless' or 'time-consuming' parameters of the input data space can be narrowed through using the information provided by exactly solving the sub-problems. In this regard, the reduced instances obtained from our approach allow to solve those instances to optimality as can be checked in the table. Moreover, we can conclude that POPMUSIC-G is faster than CPLEX alone in solving these instances.

The temporal performance of the POPMUSIC approaches are meaningfully compared to the ones exhibited by CS-SA and PSO. In this regard, although both approximate methods are able to provide the optimal solution values for the instances considered in this table, they cannot guarantee optimality. In this sense, once the POPMUSIC and POPMUSIC-G processes are over and the problem instances are reduced, those reduced instances are solved by CPLEX to optimality. This offers a benefit in terms of robustness allowing a similar quality within all iterations. In this sense, although our approaches include a heuristic component, the procedures proposed in this work are a step forward in the matheuristic concept and a step ahead over the approximate way of solving this problem.

Table 5 shows the results obtained for a representative set of instances proposed in [12]. The first column corresponds to the problem instance. In the GSPP column we report the optimal value (*Opt.*) and the required computational time (t(s.); measured in seconds). In columns POPMUSIC and POPMUSIC, we report the average objective value obtained calculated from all the executions (*Avg. Obj.*) and the average computational time (t(s.)). For the T^2S^*+PR, we report the best provided value (Obj.), relative error (Gap(%)) and running time (t(s.)) as reported in [12].

In the results shown in Table 5 it can be seen that CPLEX runs out of memory as the size of the instances becomes larger. In this sense, thanks to the POPMUSIC template, the problem can be narrowed and then the reduced problem instances can be solved to optimality.

This feature is relevant when we want to evaluate the behaviour of solution approaches for these instances, like in this case, T^2S^*+PR, where the evaluation of the performance regarding the objective value could not be done because CPLEX runs of memory without providing an upper bound. Evidently, for some of the instances we are able to improve the best solution results to date.

5 Conclusions

In this paper we have provided a POPMUSIC adaptation to the dynamic berth allocation problem. By using a given mathematical programming formulation together with the decomposition approach inherent to POPMUSIC we are able, on the one hand, to solve large scale well-known instances from the literature to optimality within less computational time than the best exact approach for this problem. On the other hand, in those cases where the mathematical model implemented in a general-purpose solver runs out of memory, our approach is able to reduce the instance and solve the reduced instance to optimality. In those cases where optimal solutions were known, the objective function values of the reduced and the overall problem instances were the same. This seems a clear advantage over the best approximate approaches known to date as the POPMUSIC approach can guarantee optimality at least for the reduced instances. Taking into account the recent related works, we can conclude that the proposed POPMUSIC approach is suitable as a resolution method for being applied either individually or included in integrated schemes where this problem is involved.

The results provided in this work highlight the successful application of POPMUSIC for solving large-sized problems. In this regard, the POPMUSIC approaches proposed in this work have a great potential for 'recognizing' relaxed constraints in the parameter space of the problem through leveraging the information obtained by solving the sub-problems. This also incorporates an explicit learning mechanism towards having an auto-adaptive control of the size of the sub-problems solved.

On the basis of the findings presented in this paper, the next stage of our research will be focused on a more indepth analysis of the POPMUSIC components such as the selection of the parts and the constitution of the sub-problems. Moreover, it may be of interest to try adapting our approach towards solving well-known allocation problems such as the ones within the knapsack problem family.

Acknowledgments This work has been partially funded by the Spanish Ministry of Economy and Competitiveness (project TIN2012-32608). Eduardo Lalla-Ruiz thanks the Canary Government for the financial support he receives through his doctoral grant. An earlier version of this paper was presented at LION8 [13].

References

1. Alvim, A., Taillard, E.: Popmusic for the world location-routing problem. Eur. J. Oper. Res. **2**(3), 231–254 (2013). doi:10.1007/s13676-013-0024-2
2. Bierwirth, C., Meisel, F.: A survey of berth allocation and quay crane scheduling problems in container terminals. Eur. J. Oper. Res. **202**, 615–627 (2010). doi:10.1016/j.ejor.2009.05.031
3. Buhrkal, K., Zuglian, S., Ropke, S., Larsen, J., Lusby, R.: Models for the discrete berth allocation problem: a computational comparison. Transp. Res. E **47**, 461–473 (2011). doi:10.1016/j.tre.2010.11.016
4. Christensen, C., Holst, C.: Berth allocation in container terminals. Master's thesis, Technical University of Denmark (2008)

5. Christiansen, M., Fagerholt, K., Nygreen, B., Ronen, D.: Chapter 4. Maritime transportation. In: C. Barnhart, G. Laporte (eds.) Transportation, Handbooks in Operations Research and Management Science, vol. 14, pp. 189–284. Elsevier (2007). doi:10.1016/S0927-0507(06)14004-9
6. Cordeau, J., Laporte, G., Legato, P., Moccia, L.: Models and tabu search heuristics for the berth-allocation problem. Transp. Sci. **39**, 526–538 (2005). doi:10.1287/trsc.1050.0120
7. Cordeau, J.F., Gaudioso, M., Laporte, G., Moccia, L.: The service allocation problem at the Gioia Tauro maritime terminal. Eur. J. Oper. Res. **176**(2), 1167–1184 (2007)
8. Hoffarth, L., Voß, S.: Liegeplatzdisposition auf einem Container Terminal - Ansätze zur Entwicklung eines entscheidungsunterstützenden Systems. In: H. Dyckhoff, U. Derigs, M. Salomon, H. Tijms (eds.) Operations Research Proceedings 1993, pp. 89–95. Springer, Berlin (1994). doi:10.1007/978-3-642-78910-6_28
9. Hu, Q.M., Hu, Z.H., Du, Y.: Berth and quay-crane allocation problem considering fuel consumption and emissions from vessels. Comput. Ind. Eng. **70**, 1–10 (2014). doi:10.1016/j.cie.2014.01.003
10. Imai, A., Nishimura, E., Papadimitriou, S.: The dynamic berth allocation problem for a container port. Transp. Res. B Methodol. **35**(4), 401–407 (2001)
11. Lalla-Ruiz, E., González-Velarde, J.L., Melián-Batista, B., Moreno-Vega, J.M.: Biased random key genetic algorithm for the tactical berth allocation problem. Appl. Soft Comput. **22**, 60–76 (2014)
12. Lalla-Ruiz, E., Melián-Batista, B., Moreno-Vega, J.M.: Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem. Eng. Appl. Artif. Intell. **25**(6), 1132–1141 (2012)
13. Lalla-Ruiz, E., Voß, S.: Towards a matheuristic approach for the berth allocation problem. Lect. Notes Comput. Sci **8426**, 218–222 (2014)
14. Li, B., Li, W.F., Voß, S.: Modeling container terminal scheduling systems as hybrid flow shops with blocking based on attributes. In: S. Voß, J. Pahl, S. Schwarze (eds.), pp. 413–434. Physica, Heidelberg (2009)
15. Maniezzo, V., Stützle, T., Voß, S. (eds.): Matheuristics - Hybridizing Metaheuristics and Mathematical Programming. Ann. Inf. Syst., vol 10. Springer, Berlin (2010)
16. Meisel, F.: Seaside Operations Planning in Container Terminals. Contributions to Management Science. Physica, Heidelberg (2010)
17. Notteboom, T.: The time factor in liner shipping services. Marit. Econ. Logist. **8**(1), 19–39 (2006)
18. de Oliveira, R.M., Mauri, G.R., Lorena, L.A.N.: Clustering search for the berth allocation problem. Expert Syst. Appl. **39**(5), 5499–5505 (2012). doi:10.1016/j.eswa.2011.11.072
19. Pei, J., Liu, X., Pardalos, P., Fan, W., Yang, S., Wang, L.: Application of an effective modified gravitational search algorithm for the coordinated scheduling problem in a two-stage supply chain. Int. J. Adv. Manuf. Technol. **70**(1-4), 335–348 (2014). doi:10.1007/s00170-013-5263-8
20. Resende, M., Pardalos, P. (eds.): Handbook of Applied Optimization. Oxford University Press, Oxford (2002)
21. Rodriguez-Molins, M., Ingolotti, L., Barber, F., Salido, M., Sierra, M., Puente, J.: A genetic algorithm for robust berth allocation and quay crane assignment. Prog. Artif. Intell., 1–16 (2014). doi:10.1007/s13748-014-0056-3
22. Sniedovich, M., Voß, S.: The corridor method: a dynamic programming inspired metaheuristic. Control. Cybern. **35**(3), 551–578 (2006). http://eudml.org/doc/209435
23. Steenken, D., Voß, S., Stahlbock, R.: Container terminal operations and operations research – a classification and literature review. OR Spectrum **26**, 3–49 (2004). doi:10.1007/s00291-003-0157-z
24. Taillard, E., Voß, S.: POPMUSIC – partial optimization metaheuristic under special intensification conditions. In: C. Ribeiro, P. Hansen (eds.) Essays and Surveys in Metaheuristics, pp. 613–629. Kluwer, Boston (2002). doi:10.1007/978-1-4615-1507-4_27
25. Ting, C.J., Wu, K.C., Chou, H.: Particle swarm optimization algorithm for the berth allocation problem. Expert Syst. Appl. **41**(4), 1543–1550 (2014). doi:10.1016/j.eswa.2013.08.051
26. Umang, N., Bierlaire, M., Vacca, I.: Exact and heuristic methods to solve the berth allocation problem in bulk ports. Transp. Res. E **54**, 14–31 (2013). doi:10.1016/j.tre.2013.03.003
27. Verstichel, J., Berghe, G.: A late acceptance algorithm for the lock scheduling problem. In: S. Voß, J. Pahl, S. Schwarze (eds.) Management, Logistik, pp. 457–478. Physica, Heidelberg (2009)
28. Xiao, L., Hu, Z.H.: Berth allocation problem with quay crane assignment for container terminals based on rolling-horizon strategy. Math. Probl. Eng. **2014** (2014). doi:10.1155/2014/845752. Article ID 845752, 11 pages
29. Zampelli, S., Vergados, Y., Schaeren, R., Dullaert, W., Raa, B.: The berth allocation and quay crane assignment problem using a CP approach. Lect. Notes Comput. Sci **8124**, 880–896 (2013). doi:10.1007/978-3-642-40627-0_64