A MEMETIC ALGORITHM FOR A CONTINUOUS CASE OF THE BERTH ALLOCATION PROBLEM

Geraldo Regis Mauri1, Larice Nogueira de Andrade2 and Luiz Antonio Nogueira Lorena3

1Federal University of Espírito Santo - UFES, Alegre, ES, Brazil
2Federal University of Triângulo Mineiro - UFTM, Uberaba, MG, Brazil
3National Institute for Space Research - INPE, São José dos Campos, SP, Brazil

Keywords: Berth allocation, Memetic algorithm, Simulated annealing, Combinatorial optimization.

Abstract: This work presents a Memetic Algorithm heuristic to solve a continuous case of the Berth Allocation Problem (BAP). The BAP deals with programming and allocating ships to berthing areas along a quay. In general, the continuous case considers that ships have different lengths and can moor anywhere along the quay. However, we consider a quay divided in berths that have limited areas and different equipments to handle the ships. So, we must to assign the ships to berths and determine the berthing time and position for each ship. We treat the ships as rectangles to be placed into a \( space \times time \) area avoiding overlaps and satisfying time window constraints. Our MA uses a Simulated Annealing (SA) as the local search mechanism, and SA is also applied in a stand alone way to solve the BAP. A two-phase heuristic is also presented to compute the berthing time and position for all of ships during MA and SA execution. Computational results are performed on a set of instances proposed in the literature and new best-known solutions are presented.

1 INTRODUCTION

The port importance grows with the increasing progress in the construction technology of great ships. Besides, due to the improved trades among the nations caused by globalization, the frequency of transports among ports also increases, mainly due to the constant economical growth and volume of the trade among several countries.

Maritime transport can be seen as a central pillar for the international trade. Approximately 80% of global market is conducted through the sea (Buhkal et al., 2011). Moreover, in 2008 the fleet of ships carrying containers had an increase in capacity of 17.3 million tons, or 11.9%, and now represents 13.6% of world total. In 2009, the world merchant fleet reached 1.19 million, an increase of 6.7% compared to January 2008, and since the beginning of the decade, the number of containers increased by 154% (UNCTAD, 2009).

Due to intense flow of ships and containers in ports, these are forced to invest heavily to accommodate the ships, deepening and widening channels and building new installations of moor, everything to minimize the service time of the ships. So, a port should operates in an efficient way (Hansen et al., 2008).

The allocation and programming of ships to berths have a primary impact on the efficiency of these operations (Imai et al., 2003). Vis and Koster (2003) present a discussion about the decision problems that appear in a port. A classification of the processes and operations of different marine terminals is presented by Steenken et al. (2004). The crucial problem in a port administration is optimizing the balance among the ships holder that demand fast services, and the economical use of the available resources into the port (Dragovic et al., 2005).

The Berth Allocation Problem (BAP) consists of optimally assigning ships to berthing areas along a quay in a port. The choice of “where” and “when” the ships shall berth is the main decision to be made in that process (Cordeau et al., 2005).

The BAP has a lot of physical and technical restrictions, among others. This makes possible to model it in different ways. Considering the spatial aspects of the berths, the BAP can be modeled as discrete and continuous (Imai et al., 2005). In the discrete case, the quay is divided into several berths and only one ship is serviced at a time in every berth, regardless of their length. In the continuous case, there is no division of the quay and thus the ships can moor at any position. Another conception for the continu-
ous case considers that the quay is divided in berths, but the big ships can occupy more than one position, thus allowing small ships to share their berth (Cordeau et al., 2005). This last approach represents the real cases of a more appropriate way, where some berths have different equipments (reaching limited areas) to operate specific type of cargo (container of various sizes, for example).

If we consider the arrival of ships, the BAP can be still treated as static or dynamic (Imai et al., 2001). The static case assumes all ships already in port for the service, while the dynamic case allows ships to arrive at any time (estimated in advance). In recent works, we have notice that the term “dynamic” has been unusual, because in practice, the static case is not close to real situations. Then, the remainder of this paper will denotes the “dynamic and continuous case” only as continuous case.

This paper solves a continuous case of the BAP by means of an effective Memetic Algorithm (MA) using a Simulated Annealing (SA) as the local search mechanism. SA is also applied in a stand alone way, and both SA and MA use a two-phase heuristic to compute the berthing time and position for all of ships presenting results that outperform those obtained by another heuristic approach reported in the literature.

The remainder of this paper is organized as follows. The next section reports a brief literature review. Section 3 presents details about the BAP. Our MA, SA and two-phase heuristic are described in Section 4. Computational results are presented in Section 5, followed by conclusions in Section 6.

2 BRIEF LITERATURE REVIEW

Initial works about the BAP appear at the end of the 80s, but these works have been gaining focus in the last 10 years.

Approaches based on using heuristic alternatives to solve the BAP are the most explored, and we believe that such methods have been predominating for allowing the inclusion of several constraints in a “soft way”.

Imai et al. (2001) present a method based on Lagrangian relaxation of the original problem. Two years later, Imai et al. (2003) upgraded their approach considering different service priorities between the ships. Furthermore, the authors proposed a Genetic Algorithm as a method of solution. The same authors also treated a continuous case (see Imai et al. (2005)).

Cordeau et al. (2005) present a Tabu Search based heuristic for solve two different models for a discrete case of the BAP. They also present a TS heuristic for a continuous case of the problem. They compared their TS with a truncated branch-and-bound applied to an exact formulation. Comparisons were made for the discrete case. For the continuous approach, the TS was able to solving more realistic instances than those previously considered by other authors. The proposed TS could handles various features of real-life problems, including time windows and favorite and acceptable berthing areas. The objective function treated a weighted sum of the ship’s service times.

Mauri et al. (2008a) proposed an approach based on the application of Simulated Annealing for solving the discrete case of BAP. The authors treat the problem as a Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW). The computational results improved those obtained by CPLEX and Tabu Search proposed by Cordeau et al. (2005). Mauri et al. (2008b) solves the discrete case of the BAP by using a hybrid method called PTA/LP, which uses the Population Training Algorithm with a Linear Programming model through a Column Generation technique. The results improved those shown in their previous work.

Giallombardo et al. (2010) present two models based on quadratic and linear programming to represent a discrete case of the BAP. Moreover, the authors use a Tabu Search and a mathematical programming technique to solve instances based on real data.

Recently, Buhrkal et al. (2011) reviewed some approaches about BAP and described three models for the discrete berth allocation problem. The authors proven the optimal solutions for some instances presented by Cordeau et al. (2005) for the discrete case of BAP.

3 BERTH ALLOCATION

As mentioned before, the Berth Allocation Problem (BAP) consists of optimally assigning ships to berthing areas along a quay in a port respecting several physical and technical restrictions.

Related to the berth position, there exist constraints concerning to the depth of the water (allowable draft) and to the maximum distance from the most favorable location along the quay, considering the ship’s length and the location of the outbound and inbound containers.

Some ports have hard allowable draft constraints since they are located in geographical regions subject to meaningful tide variation. In such cases, the ships only can berth in known time windows of about two hours around the high tide.

Besides this time window constraint, there exist the time windows for completion time of ship service-
ing (Cordeau et al., 2005). The handling time of a ship depends on its berthing point and it is a function of the distance from the berth to the pick-up and delivery area of containers stored in the port yard. This dependency strongly affects the performance of the port (Cordeau et al., 2005).

Managers want to minimize both port and user costs, which are related to service time. The objective of the BAP is usually to minimize the total service time of all ships. Since ships do not have equal importance, a weighted sum of the ship service times may better reflect the management practice of some ports. The weights in this sum can represent a pricing scheme (demurrage) over delayed operations, the number of container moves, and the urgent need of a given load. Penalty terms can be explicitly included in the objective function, for example, the demurrage cost when the service time of a ship exceeds the contracted value (see Barros et al. (2011), for example).

As reported by Cordeau et al. (2005), the BAP can be represented in a two-dimensional area (see Figures 1 and 2) considering the ships as rectangles whose dimensions are length, including a safety margin, and handling time (space \times time area). So, the BAP can be defined by placing these rectangles in the port area without overlapping and satisfying several constraints.

The spatial dimension is ignored in the discrete case of the BAP, and the berths can be described as fixed length segments, or simple points. For the continuous case, we consider that the quay is divided into berths (as explained in Section 1), but now the spatial dimension is not ignored. We will denote these approaches as BAP-D and BAP-C, respectively. For the BAP-C, a ship may occupy part of neighboring berths (see Cordeau et al. (2005)).

Cordeau et al. (2005) considered a quay divided into berths with two parts, left and right. So, each berth \( k \) has two neighbors, the right part of the berth \( k - 1 \) and the left part of the berth \( k + 1 \). Discontinuous segments must be also considered, representing the initial and final berths and also natural obstacles, as sharp curves, for example. In these cases, the berths have only one neighbor and they are not divided (see berths 1, 7, 8, 9 and 13 in Figure 1).

Figure 1 presents an optimal solution (cost = 1409) for the instance i01 (one of those used on our computational experiments - see Section 5) considering the BAP-D approach. In this figure, we used the colors red and blue to enhance visualization. Then, each rectangle represents a ship, and the blue ones are assigned to the nearest blue berth. This is also true for the red ones. Marked green lines indicate discontinuities and solid and dotted black lines indicate the different berths and their parts (left and right), respectively. Bottom and top filled boxes indicate the start and the end of availability time for the berths. We can see that there are overlapping of ships, and the discontinuous berths are not respected.

A best-known solution (cost = 1613 - found by our MA described in Section 4) for this same instance (i01) considering the continuous case (BAP-C) is presented in Figure 2. No overlapping and violations of the discontinuities are observed in this case.

Cordeau et al. (2005) reported that BAP-D can be regarded as a relaxation of the BAP-C, because a solution for the BAP-D can violate the spatial constraints (compare Figures 1 and 2). So, we notice that a solution for the BAP-D can be seen as a lower bound for the BAP-C (Imai et al., 2005). In this paper, we will treat only the continuous case (BAP-C) aiming to minimize the sum of the times while the ships stayed into the port (service times), i.e., the elapsed time since the ships incoming, berthing and handling. Figure 3 illustrates the service time for a ship \( i \) assigned to a hypothetical berth 3.

A MEMETIC ALGORITHM FOR A CONTINUOUS CASE OF THE BERTH ALLOCATION PROBLEM

Figure 1: Optimal solution for BAP-D: i01 = 1409.

Figure 2: Best-known solution for BAP-C: i01 = 1613.
vance, the following data:

- \( n \): number of ships;
- \( m \): number of berths;
- \( a_i \): arrival time of ship \( i \);
- \( s_i \): start of availability time for berth \( i \);
- \( f_i \): start of availability time for berth \( i \);
- \( e_i \): end of availability time for berth \( i \);
- \( P_k^i \): position for berth \( k \) (including part of its right neighbor);
- \( P_k^e \): position for berth \( k \) (including part of its left neighbor);
- \( L_k^i \): neighbor at right side for berth \( k \) (0 if none);
- \( L_k^e \): neighbor at left side for berth \( k \) (0 if none).

Using these data, we must assign the ships to berths determining the berthing time and position for each ship \( i \) \((i = 1, \ldots, n)\). We will denote them as \( T_i \) and \( P_i \) respectively. These variables can be seen as the coordinates for the left lower corner point of the rectangles (see Figure 3).

Several mathematical models for both BAP-D and BAP-C can be found in the literature. See for example Imai et al. (2001), Nishimura et al. (2001), Cordeau et al. (2005) and Buhrkal et al. (2011). However, we did not find any model for the continuous case (BAP-C) that considers all of the features previously described, as discontinuities, the use of neighboring berths and different berths capacity (different handling times). A comparison among some recent models is presented by Buhrkal et al. (2011).

### 4 MEMETIC ALGORITHM

In this section, we present our MA algorithm for the BAP-C.

Evolutionary algorithms have been applied successfully for solve several classes of combinatorial optimization problems. Memetic Algorithms (MAs) are evolutionary algorithms which can be seen as Genetic Algorithms (GAs) that uses a local search procedure to improve the search refining individuals (Moscato, 1999).

The “meme” suggests a cultural evolution within a lifetime, and MA uses this concept by applying a local search to represent a learning to the individuals.

The MAs use evolutionary operators to find promising regions on the search space applying a local search procedure to intensify the investigation in these regions. According to Moscato and Norman (1992), the basic idea of the MAs consists of exploring the neighborhood of the solutions obtained by a GA searching to local optimal.

The recombination mechanism (crossover) allows that the mixing of information from one generation can be transmitted to its offspring, and the mutation introduces innovation in the population. More details about the MAs are reported by Moscato (1999).

#### Algorithm 1: Memetic Algorithm - MA.

1. **Input:** \(#\text{gen}, \#\text{pop}, \#\text{eli}, \#\text{cro}, \#\text{mut}\) and \#\text{ls}.
2. \(\text{curPop} \leftarrow \text{INITIAL-POPULATION}(\#\text{pop})\)
3. \(\text{bstInd} \leftarrow \text{the best individual in curPop}\)
4. \(\text{newPop} \leftarrow \{\}\)
5. \(\text{for } i = 1 \text{ to } \#\text{gen} \text{ do}\)
6. \(\text{ for } j = 1 \text{ to } \#\text{cro} \text{ do}\)
7. \(\text{ if } j \leq \#\text{cro} \text{ do}\)
8. \(\text{ p1 } \leftarrow \text{SELECT(one from the best } \#\text{eli}\% \text{ in curPop)}\)
9. \(\text{ p2 } \leftarrow \text{SELECT(one from all in curPop)}\)
10. \(\text{ of f } \leftarrow \text{ CROSSOVER} (p_1, p_2)\)
11. \(\text{ of f } \leftarrow \text{ LOCAL-SEARCH} (\text{off, } \#\text{ls})\)
12. \(\text{ newPop } \leftarrow \text{ADD} (\text{off})\)
13. \(\text{ end}\)
14. \(\text{ for } j = 1 \text{ to } \#\text{mut} \text{ do}\)
15. \(\text{ ind } \leftarrow \text{SELECT(one from newPop)}\)
16. \(\text{ MUTATION} (\text{ind})\)
17. \(\text{ end}\)
18. \(\text{ bstInd } \leftarrow \text{BEST} (\text{bstInd, ind})\)
19. \(\text{ DISCARD} (\text{curPop})\)
20. \(\text{ curPop } \leftarrow \text{the best } \#\text{pop} \text{ individuals in newPop}\)
21. \(\text{ newPop} \leftarrow \{\}\)
22. \(\text{ end}\)
23. **Output:** \(\text{bstInd}\)

Our MA contains some user-controlled parameters, and we will denote them as: \#\text{gen}, \#\text{pop}, \#\text{eli}, \#\text{cro}, \#\text{mut}\) and \#\text{ls}. \#\text{gen} is the number of generations for the algorithm converging; \#\text{pop} is the population size of the population.
size (number of individuals); \#el is a percentage of the best individuals in a population (elite); \#cro and \#mut are the number of crossover and mutation performed at each generation; and \#l is the “deep” of the local search.

First, \#pop individuals (solutions) are randomly created to form an initial population. \#cro crossovers are performed between two parents randomly selected from the current population: one elite individual and any other. A local search is performed over the resulting offspring, and it is added into a new population. Then, \#mut mutations are applied over individuals randomly selected from the new population. The best individual in the new population is stored, the current population is discarded and the new population becomes the current one. This process is repeated for \#gen generations. Our MA pseudo-code is presented in Algorithm 1.

We now describe the main elements of our MA implementation for the BAP-C.

Individual Representation. An individual represents one possible solution for the BAP-C and it is described as a matrix with the lines indicating the berths and the columns presenting the sequence of ships to be handling at each berth. Figure 4 illustrates an individual (solution) for a hypothetical problem with 4 berths and 14 ships.

```
Berth 1 | 4 | 10 | 6 | 13 |
Berth 2 | 2 | 9  |
Berth 3 | 7 | 1  | 3 | 8  | 12 |
Berth 4 | 11| 14 | 5 |
```

Figure 4: Individual representation.

Initial Population. A random individual can be constructed through a simple and balanced heuristic. Initially, the ships are organized by incoming order on port \((a_i)\) and distributed to the berths in a random way. In this process, the selected berth must always be able to assist the selected ship. This procedure ensures that each ship will be assigned to a berth that must be able to attend it, i.e., the berth length is sufficient to receive the ship and the berth’s equipments are suitable to operate the type of cargo into the ship. So, this procedure is repeated for \#pop times forming an initial population. This procedure does not guarantee that the berthing time \((T_i)\) and position \((P_i)\) for the ship \(i (i = 1, \ldots, n)\) present no overlapping on time and space dimensions. This strategy is based on the distribution heuristic presented by Mauri et al. (2008a,b) and the FCFS-G heuristic presented by Cordeau et al. (2005).

Computing Berthing Times and Positions. After generating a random individual (solution), we must define the berthing time and position for all of ships. We present a two-phase heuristic to determine these values.

- **Phase 1:** Berthing time and position for all of ships assigned to a specific berth are set to initial values according to Algorithm 2. At this moment, the berthing times are set to equal the arrival times of the ships, if the berth is available. The berthing positions are set to equal the start position of the berth. This phase is applied to give initial values for \(P_i\) and \(T_i\), that are updated in phase 2. This procedure is based on the ones presented by Mauri et al. (2008a,b), Pacheco et al. (2010) and Ribeiro et al. (2011) adding the position defining.

```
Algorithm 2: Two-phase heuristic - phase 1.
1 Input: berth k
2 for each ship i assigned to k do
3 \(T_i \leftarrow \max (a_i, t_i)\)
4 \(P_i \leftarrow P_{i-1} - L_i\)
5 end
```

- **Phase 2:** The spatial distribution for the ships to the berths are updated and improved. This procedure updates the berthing times and positions considering a simple idea: if some overlap is detected for each ship, its berthing time is delayed until eliminate all of them. If a berth has no neighbor at the left side, all of ships assigned to it have the berthing position set to equal the start position of the berth, i.e., all of ships (rectangles) are aligned at left. If a berth has no neighbor at the right side, all of ships assigned to it have the berthing position set to equal the final position of the berth minus the ship length, i.e., all of ships are aligned at right. Finally, if a berth has neighbors on both sides, we try to fit the ships among the ones assigned to the two neighbor berths. This procedure is described in Algorithm 3.

Figures 5 and 6 illustrate a randomly generated individual after running only phase 1 and after phases 1 and 2 of our two-phase heuristic.

In the first case (Figure 5), only the phase 1 of our heuristic was performed, and we can note that overlapping were not treated. In the second case (Figure 6), both phases were computed and no overlapping is presented. However, we can observe that some ships violate the deadlines for some berths (see berths 2, 3, 4, 8 and 9), and these violations must be eliminated by our MA through a penalized objective function (see expression 1).
Algorithm 3: Two-phase heuristic - phase 2.

1. **Input:** berth $k$
2. **for each** ship $i$ **assigned to** $k$ **do**
   3. **if** $k_i = 0$ **then**
   4. $P_i ← P_i$
   5. **for each** ship $j$ **assigned to** $k + 1$ **do**
   6. **if** $i$ **overlaps** $j$ **then**
   7. $T_i ← \max(T_i, T_j + t_i^{k+1})$
   8. **end**
   9. **end**
  10. **else**
  11. **if** $P_i = 0$ **then**
  12. $P_i ← P_i  - z_i$
  13. **for each** ship $j$ **assigned to** $k - 1$ **do**
  14. **if** $i$ **overlaps** $j$ **then**
  15. $T_i ← \max(T_i, T_j + t_i^{k-1})$
  16. **end**
  17. **end**
  18. **else**
  19. $P_i ← P_i$
  20. **while** $\exists j$ **assigned to** $k - 1$ **overlapping** $i$ **do**
  21. **if** $(P_j + z_j > P_i)$ **and** $(P_j + z_j < P_i)$ **then**
  22. $P_i ← P_i  + z_i$
  23. **else**
  24. $P_i ← \max(P_i, T_j + t_i^{k-1})$
  25. $P_i ← P_i  - z_i$
  26. **end**
  27. **while** $\exists j$ **assigned to** $k + 1$ **overlapping** $i$ **do**
  28. $T_i ← \max(T_i, T_j + t_i^{k+1})$
  29. **while** $\exists i$ **assigned to** $k - 1$ **overlapping** $j$ **do**
  30. $T_j ← \max(T_j, T_i + t_j^{k-1})$
  31. **end**
  32. **end**
  33. **end**

**Figure 5:** Random solution after phase 1.

**Figure 6:** Random solution after phases 1 and 2.

**Individual Evaluation.** We may notice that our two-phase heuristic guarantees that no overlapping will occur and no ship will exceed the spatial limit of the berths. However, our solutions can still present violations on the limit time to complete the service of the ships $(b_i, \forall i = 1, \ldots, n)$ and on the end of availability time of the berths $(e_k, \forall k = 1, \ldots, m)$. So, we define a function to compute the cost of an individual considering a weight $\lambda$ to penalty theses violations (see expression (1)). This function is composed by three terms representing the service times for the ships (sum of the times when the ships stayed in port) and the sum of violating deadlines for the ships and berths respectively. The fitness measure used to evaluate the individuals was $1/f(I)$.

$$f(I) = \left\{ \sum_{k \in M} \sum_{i \in N} (T_i + t_i^k - a_i) + \lambda \sum_{k \in M} \sum_{i \in N} \max(0, T_i + t_i^k - b_i) + \lambda \sum_{k \in M} \sum_{i \in N} \max(0, T_i + t_i^k - e_k) \right\} \quad (1)$$

**Crossover.** A one-point based crossover was implemented. Given two parents $p_1$ and $p_2$, a number of berths $x$ ($0 < x < m$) is randomly selected and then the first $x$ berths from $p_1$ are inserted into the offspring $off$. Then, the last $m - x$ berths from $p_2$ are inserted into $off$ removing the duplicated ships, and the ships that have not been assigned to any berth
are randomly allocated at the last \( m - x \) berths in \( off \).
The ships assigned to each of the last \( m - x \) berths are
organized by incoming order on port \((a)\) and the
two-phase heuristic is applied only to these berths in
\( off \). Finally, function \( I \) is used to compute the cost
\( f(\text{off}) \) and fitness for \( off / f(\text{off}) \). Figure 7
illustrates our crossover.

**Mutation.** Three different procedures explored in
previous works (see Mauri et al. (2008a), Pacheco
et al. (2010) and Ribeiro et al. (2011)) were used
to compose a neighborhood structure: Re-order
ships, Re-allocate ship and Swap ships. We will
denote them as \( \omega \), \( \omega \) and \( \omega \) respectively. \( \omega \) is
obtained by randomly choosing two ships of the
same berth, and swapping them; \( \omega \) is obtained by
inserting a ship \( i \) chosen at random from some berth
into another berth in a position chosen at random;
and \( \omega \) is obtained by swapping two ships from
two different berths, all chosen at random. So, a mutation
is defined by applying one of these procedures
(randomly selected) over an individual followed
by performing the two-phase heuristic on the modi-
plied berths. Finally, the individual’s fitness is updated.

**Local Search.** A traditional Simulated Annealing
(SA) presented in previous works (see Ribeiro et al.
(2011)) was used as the local search mechanism for
our Memetic Algorithm (MA). The neighborhood
structure for our SA were composed by selecting
(at random) one of the three procedures adopted in the
Mutation. We must emphasize that low values for the
SA parameters were used into our MA to make only a
fast local search. These parameters values were: \( \alpha = 0.95 \), \( T_{\text{start}} = 10 \), \( T_{\text{frozen}} = 0.01 \) and \( S_{\text{max}} = \#l \) (de-
defined in the next section).

### 5 COMPUTATIONAL RESULTS

This section presents a comparative evaluation of our
Memetic Algorithm (MA) and Simulated Annealing
(SA) applied in a stand alone way. Both heuristics
were applied 10 times to each instance, and to be fair,
we imposed a time limit as unique stopping criterion
for them: 120 seconds for each instance, i.e., the same
time used by Tabu Search (TS) proposed by Cordeau
et al. (2005).

The SA and MA heuristics have stopping conditions
based on the temperature and on the number of
generations, respectively. If they reach their stopping
conditions before the time limit, we restart the heuris-
tic from the best solution identified during the search.
This process is repeated until the heuristic reaches the
time limit.

The SA and MA heuristics were coded in C++ and ran on a
PC with Intel Core i3 processor of 2.66 GHz with 4 GB of
RAM Memory under Windows 7 operating system. Cordeau et al. (2005) had use
a Sun workstation (900 MHz) to run their TS. Our algo-
rithms were tested on a BAP instances set randomly
generated by Cordeau et al. (2005). This set is com-
posed by 30 different instances with 60 ships and 13
berths, and it was provided to us by the authors in a
previous situation.

The parameters of SA and MA were chosen empirically after running them over three randomly
selected instances. We ran SA and MA five times for
each parameter setting, and the setting yielding the
best average results was chosen. So, the final values
for the SA parameters were: \( T_{\text{start}} = 15000 \), \( T_{\text{frozen}} =
0.01 \), \( \alpha = 0.975 \), \( S_{\text{max}} = 1000 \). The parameters for
our MA were: \( #\text{pop} = 30 \), \( #\text{el} = 10 \), \( #\text{cro} = 25 \), \( #\text{mut} = 5 \) and \( #\text{ls} = 7 \). The parameters used by SA (as a
local search) inside the MA were defined in Section
4.

A special attention was given to the weight \( \lambda \), be-
cause we note that a large value for this weight often
traps the search in a locally suboptimal solution. So,
the weight value was set to 10 (\( \lambda = 10 \)).

Table 1 presents a detailed comparison between
our SA and MA heuristics. Five different columns
are used for all results: the instance name, the best so-

| Instance | Time (s) | | Time (s) |
|----------|---------| |---------|
| b          | bst | ave | ave | bst | ave | ave | bst | ave | ave |
| 401         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 402         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 403         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 404         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 405         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 406         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 407         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 408         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 409         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 410         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 411         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 412         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 413         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 414         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 415         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 416         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 417         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 418         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 419         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 420         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 421         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 422         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 423         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 424         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 425         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 426         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 427         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 428         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 429         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
| 430         | 1469.60 | 21.95 | 58.45 | 1469.60 | 21.95 | 58.45 |
olution found ($f(Y) \rightarrow bst$), the average solution found (avg), the computational time (bks) to reach the best-known so far (TS by Cordeau et al. (2005)) and the computational time to find the best solution for each algorithm ($\text{Times}(s) \rightarrow bst$).

In Table 1, we can note that MA found ten best solutions and four worse than SA. The others sixteen solutions were equal. Despite the differences, we notice that MA is slightly better, but in general, the solutions of SA and MA are very close. This fact can indicate that our solutions are probably next to the optimal ones. We confirmed this situation in Table 3. The average values (last line) are also close, and we emphasize that our heuristics find the same solutions of TS proposed by Cordeau et al. (2005) using less than 30 seconds.

Table 2: Comparison with the TS of Cordeau et al. (2005).

<table>
<thead>
<tr>
<th>Instance</th>
<th>TS</th>
<th>MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>bst dev (%)</td>
<td>bst dev (%)</td>
<td></td>
</tr>
<tr>
<td>i01</td>
<td>1706 5.77</td>
<td>1659 161</td>
</tr>
<tr>
<td>i02</td>
<td>1355 2.19</td>
<td>1326 0</td>
</tr>
<tr>
<td>i03</td>
<td>1268 5.21</td>
<td>1242 0.65</td>
</tr>
<tr>
<td>i04</td>
<td>1440 3.45</td>
<td>1392 0</td>
</tr>
<tr>
<td>i05</td>
<td>1352 5.21</td>
<td>1285 0</td>
</tr>
<tr>
<td>i06</td>
<td>1565 7.12</td>
<td>1461 0</td>
</tr>
<tr>
<td>i07</td>
<td>1389 4.20</td>
<td>1333 0</td>
</tr>
<tr>
<td>i08</td>
<td>1519 6.60</td>
<td>1435 0</td>
</tr>
<tr>
<td>i09</td>
<td>1713 3.76</td>
<td>1651 0</td>
</tr>
<tr>
<td>i10</td>
<td>1411 2.92</td>
<td>1371 0</td>
</tr>
<tr>
<td>i11</td>
<td>1696 8.93</td>
<td>1574 1.09</td>
</tr>
<tr>
<td>i12</td>
<td>1629 5.99</td>
<td>1541 0.26</td>
</tr>
<tr>
<td>i13</td>
<td>1519 4.83</td>
<td>1457 0.55</td>
</tr>
<tr>
<td>i14</td>
<td>1369 6.62</td>
<td>1284 0</td>
</tr>
<tr>
<td>i15</td>
<td>1455 6.83</td>
<td>1362 0</td>
</tr>
<tr>
<td>i16</td>
<td>1715 13.73</td>
<td>1508 0</td>
</tr>
<tr>
<td>i17</td>
<td>1322 0.92</td>
<td>1310 0</td>
</tr>
<tr>
<td>i18</td>
<td>1594 4.94</td>
<td>1529 0.66</td>
</tr>
<tr>
<td>i19</td>
<td>1673 6.42</td>
<td>1572 0.76</td>
</tr>
<tr>
<td>i20</td>
<td>1450 1.54</td>
<td>1438 0.70</td>
</tr>
<tr>
<td>i21</td>
<td>1565 5.89</td>
<td>1478 0</td>
</tr>
<tr>
<td>i22</td>
<td>1618 9.03</td>
<td>1484 0</td>
</tr>
<tr>
<td>i23</td>
<td>1539 8.00</td>
<td>1425 0</td>
</tr>
<tr>
<td>i24</td>
<td>1425 4.86</td>
<td>1365 0.44</td>
</tr>
<tr>
<td>i25</td>
<td>1590 2.85</td>
<td>1546 0</td>
</tr>
<tr>
<td>i26</td>
<td>1567 6.24</td>
<td>1475 0</td>
</tr>
<tr>
<td>i27</td>
<td>1458 7.52</td>
<td>1356 0</td>
</tr>
<tr>
<td>i28</td>
<td>1550 4.31</td>
<td>1404 1.21</td>
</tr>
<tr>
<td>i29</td>
<td>1415 5.75</td>
<td>1340 0.15</td>
</tr>
<tr>
<td>i30</td>
<td>1621 7.21</td>
<td>1512 0</td>
</tr>
</tbody>
</table>

Table 3: Comparison with the optimal solutions for BAP-D.

<table>
<thead>
<tr>
<th>Instance</th>
<th>OPT</th>
<th>BAP-D</th>
<th>BAP-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>bst (%)</td>
<td>bst (%)</td>
<td>bst (%)</td>
<td></td>
</tr>
<tr>
<td>i01</td>
<td>1499 1798 1639 1563</td>
<td>21.06 16.32 14.44</td>
<td></td>
</tr>
<tr>
<td>i02</td>
<td>1261 1355 1326 1326</td>
<td>7.45 5.15 5.15</td>
<td></td>
</tr>
<tr>
<td>i03</td>
<td>1129 1286 1242 1234</td>
<td>13.91 10.01 9.30</td>
<td></td>
</tr>
<tr>
<td>i04</td>
<td>1302 1440 1392 1392</td>
<td>10.60 6.91 6.91</td>
<td></td>
</tr>
<tr>
<td>i05</td>
<td>1207 1352 1265 1261</td>
<td>7.15 4.65 4.65</td>
<td></td>
</tr>
<tr>
<td>i06</td>
<td>1261 1565 1461 1461</td>
<td>24.11 15.86 15.86</td>
<td></td>
</tr>
<tr>
<td>i07</td>
<td>1279 1398 1333 1333</td>
<td>8.60 4.22 4.22</td>
<td></td>
</tr>
<tr>
<td>i08</td>
<td>1299 1318 1425 1425</td>
<td>16.94 9.70 9.70</td>
<td></td>
</tr>
<tr>
<td>i09</td>
<td>1444 1713 1651 1651</td>
<td>18.63 14.34 14.34</td>
<td></td>
</tr>
<tr>
<td>i10</td>
<td>1273 1411 1371 1371</td>
<td>16.72 13.01 13.01</td>
<td></td>
</tr>
<tr>
<td>i11</td>
<td>1368 1699 1574 1574</td>
<td>23.98 15.06 15.06</td>
<td></td>
</tr>
<tr>
<td>i12</td>
<td>1325 1429 1417 1427</td>
<td>22.94 16.30 16.30</td>
<td></td>
</tr>
<tr>
<td>i13</td>
<td>1360 1519 1457 1449</td>
<td>11.69 7.13 7.13</td>
<td></td>
</tr>
<tr>
<td>i14</td>
<td>1233 1389 1284 1287</td>
<td>11.03 4.14 4.14</td>
<td></td>
</tr>
<tr>
<td>i15</td>
<td>1295 1455 1382 1362</td>
<td>12.36 5.17 5.17</td>
<td></td>
</tr>
<tr>
<td>i16</td>
<td>1364 1715 1508 1508</td>
<td>25.73 10.56 10.56</td>
<td></td>
</tr>
<tr>
<td>i17</td>
<td>1283 1322 1310 1310</td>
<td>3.04 2.10 2.10</td>
<td></td>
</tr>
<tr>
<td>i18</td>
<td>1345 1594 1529 1519</td>
<td>16.51 13.68 12.94</td>
<td></td>
</tr>
<tr>
<td>i19</td>
<td>1382 1673 1572 1573</td>
<td>22.36 15.06 15.06</td>
<td></td>
</tr>
<tr>
<td>i20</td>
<td>1328 1450 1438 1428</td>
<td>9.19 8.28 7.53</td>
<td></td>
</tr>
<tr>
<td>i21</td>
<td>1341 1565 1478 1481</td>
<td>16.70 10.22 10.44</td>
<td></td>
</tr>
<tr>
<td>i22</td>
<td>1326 1518 1484 1484</td>
<td>22.02 11.92 11.92</td>
<td></td>
</tr>
<tr>
<td>i23</td>
<td>1266 1539 1425 1425</td>
<td>21.56 12.56 12.56</td>
<td></td>
</tr>
<tr>
<td>i24</td>
<td>1260 1425 1365 1359</td>
<td>13.10 8.33 7.86</td>
<td></td>
</tr>
<tr>
<td>i25</td>
<td>1376 1590 1546 1546</td>
<td>15.55 12.35 12.35</td>
<td></td>
</tr>
<tr>
<td>i26</td>
<td>1318 1567 1475 1475</td>
<td>18.89 11.91 11.91</td>
<td></td>
</tr>
<tr>
<td>i27</td>
<td>1261 1458 1356 1356</td>
<td>15.62 7.53 7.53</td>
<td></td>
</tr>
<tr>
<td>i28</td>
<td>1359 1500 1504 1486</td>
<td>14.05 10.67 9.35</td>
<td></td>
</tr>
<tr>
<td>i29</td>
<td>1280 1415 1340 1338</td>
<td>10.55 4.69 4.69</td>
<td></td>
</tr>
<tr>
<td>i30</td>
<td>1344 1621 1512 1512</td>
<td>20.61 12.50 12.50</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Comparison with the TS of Cordeau et al. (2005).

<table>
<thead>
<tr>
<th>Instance</th>
<th>TS</th>
<th>SA</th>
<th>MA</th>
<th>BT</th>
<th>SA</th>
<th>MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg</td>
<td>1518.77</td>
<td>1516.77</td>
<td>1439.50</td>
<td>1436.36</td>
<td>16.0</td>
<td>16.1</td>
</tr>
</tbody>
</table>

Looking at Table 3, it is interesting to highlight the gaps over the optimal solutions for the BAP-D. We can verify that our MA solutions are relatively close to the optimal, because the average gap was of 9.8%. We may also note a significant reduction over the solutions provided by the Tabu Search proposed by Cordeau et al. (2005).

6 CONCLUSIONS

In this paper we have developed a Memetic Algorithm (MA) to solve a continuous case of the Berth Allocation Problem (BAP).

MA was employed by using a Simulated Annealing (SA) as the local search mechanism and a two-phase heuristic to compute the berthing time and po-
osition for all of ships. SA was also applied in a stand alone way.

On test instances, both SA and MA yield good results and outperform the rather good Tabu Search of Cordeau et al. (2005). Our results show a relative superiority of the MA, but SA also found good solutions.

The integrated use of MA, SA and the two-phase heuristic results in a powerful approach to solve the continuous case of the BAP, providing good solutions in low computational times.

MA showed to be extremely efficient, presenting small gaps over the best-known lower bounds and suggesting that our solutions are probably close to optimal.

So, considering that the continuous case of the BAP represents real situations in a more appropriate way, and considering that BAP has a significant impact on the efficiency of the marine terminals, a minimal reduction on the service times may reflects a gain and/or economy of millions of dollars.

ACKNOWLEDGEMENTS

The authors acknowledge Espírito Santo Research Foundation - FAPES (process 45391998/09) and National Council for Scientific and Technological Development - CNPq (processes 300747/2010-1, 300692/2009-9 and 470813/2010-5) for their financial support.

REFERENCES


