A Hybrid Column Generation Approach for the Berth Allocation Problem

Geraldo R. Mauri[1,3], Alexandre C. M. Oliveira[2], and Luiz A. N. Lorena[3] *

[1] Federal University of Espírito Santo - UFES, Brazil
[2] Federal University of Maranhão - UFMA, Brazil
[3] National Institute for Space Research - INPE, Brazil
mauri@lac.inpe.br,acmo@deinf.ufma.br,lorena@lac.inpe.br

Abstract. The Berth Allocation Problem (BAP) consists on programming and allocating ships to berthing areas along a quay. The BAP is modeled as a vehicle routing problem and a recently proposed evolutionary hybrid method denominated PTA/LP is used to solve it. The PTA/LP combines the Population Training Algorithm with Linear Programming to generate improving incoming columns in a column generation process. The computational results are obtained for a set of instances proposed in literature and new best known solutions are presented.

1 Introduction

The programming and allocation of ships to berths have a primary impact in the efficiency of the port operations [1]. A discussion about the decision problems that appear in a port is presented in [2].

The Berth Allocation Problem - BAP consists of optimally assigning ships to berthing areas along a quay in a port. The main decision to be made in that process accomplishes the choice of "where" and "when" the ships shall berth [3]. Managers want to minimize both port and user costs, which are related to the ships' service time. The BAP objective is usually to minimize the total service time of all ships.

The BAP can be modeled as a discrete problem considering the quay as a finite set of berths. In this case, the berths can be described as fixed length segments, or points if the spatial dimension is ignored [1, 3]. Continuous models consider that ships can berth anywhere along the quay, where ships are of different lengths and the quay capacity varies dynamically.

In this paper, the problem is treated in discrete form considering the minimization of the time spent by ships arriving in a port, allocating and programming the ships mooring to berths, aiming to reduce the permanence time for ships inside the port. The remainder of the paper is organized as follows. Section 2 presents a brief literature review. The problem modeling is presented in Section 3. Section 4 describes the proposed model and the methods used to solve the BAP. Computational results are presented in Section 5, and the conclusions are summarized in Section 6.

* The authors acknowledge FAPESP and CNPq by partial research support.

2 Literature Review

Cordeau et al. [3] presents a Tabu Search based heuristic to solve two different models for a discrete case of BAP. Only small instances could be solved optimally and the proposed Tabu Search always yields an optimal solution. The proposed heuristics could handle the various features of real-life problems, including time windows and favorite and acceptable berthing areas. The objective function could easily accommodate a weighted sum of the ship's service times.

Filho and Lorena [4] applied a heuristic column generation approach to graph coloring. They describe the principles of their Constructive Genetic Algorithm (CGA) and give a column generation formulation for the problem. The CGA is used to generate the initial columns and also to solve the sub-problems. The column generation is performed as long as the CGA finds columns with negative reduced costs. The master problem is solved by CPLEX [5].

Recently, Puchinger and Raidl [6] proposed new integer linear programming formulations for the three-stage two-dimensional bin packing problem. Based on these formulations, a branch-and-price algorithm was developed with a fast column generation performed by applying a hierarchy of four methods: a greedy heuristic, an evolutionary algorithm, a restricted pricing problem using CPLEX, and finally the complete pricing problem also using CPLEX.

3 BAP modeling

This work considers the Berth Allocation Problem (BAP) modeled as a Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW) [7, 3] (discrete formulation). The ships are seen as customers, and the berths as depots at which one vehicle is located. There are m "vehicles", one for each depot, and each vehicle starts and finishes its tour at its depot. The ships are modeled as vertices in a multi-graph and every depot is divided into an origin and a destination vertices. The time windows can be imposed on every vertex, and its correspond to the availability period of the berth at the origin and destination vertices.

The model is given by a multi-graph $G^k = (V^k, A^k)$, $\forall k \in M$ where $V^k = N \cup \{o(k), d(k)\}$ and $A^k \subseteq V^k$ x V^k. The input data are given by:

- N: set of ships, n = $|N|$;
- M: set of berths, m = $|M|$;
- t_i^k: handling time of ship i at berth k;
- a_i: arrival time of ship i;
- s^k: start of availability time of berth k;
- e^k: end of availability time of berth k;
- b_i: upper bound for service time window for ship i;
- v_i: the value (cost) of service time for ship i.

The model variables are:

- $x_{ij}^k \in \{0, 1\}$, $k \in M$, (i,j) $\in A^k$; $x_{ij}^k = 1$ if the ship j is scheduled after ship i at berth k;

- T_i^k, $k \in M$, $i \in N$: is the berthing time of ship i at berth k;
- $T_{o(k)}^k$, $k \in M$: is the starting operation time of berth k (the time when the first ship moors at the berth);
- $T_{d(k)}^k$, $k \in M$: is the ending operation time of berth k (the time when the last ship departs from the berth);
- $M_{ij}^k = \max\{b_i + t_i^k - a_j, 0\}$, $k \in M$, i and $j \in N$.

The BAP model is as follows:

Minimize:

$$Z = \sum_{i \in N} \sum_{k \in M} v_i \left[T_i^k - a_i + t_i^k \sum_{j \in N \cup \{d(k)\}} x_{ij}^k \right] \tag{1}$$

Subject to:

$$\sum_{k \in M} \sum_{j \in N \cup \{d(k)\}} x_{ij}^k = 1 \quad \forall i \in N \tag{2}$$

$$\sum_{j \in N \cup \{d(k)\}} x_{o(k)j}^k = 1 \ \forall k \in M \tag{3}$$

$$\sum_{i \in N \cup \{o(k)\}} x_{i,d(k)}^k = 1 \ \forall k \in M \tag{4}$$

$$\sum_{j \in N \cup \{d(k)\}} x_{i,j}^k - \sum_{j \in N \cup \{o(k)\}} x_{j,i}^k = 0 \ \forall k \in M, \forall i \in N \tag{5}$$

$$T_i^k + t_i^k - T_j^k \leq (1 - x_{i,j}^k) M_{i,j}^k \ \forall k \in M, \forall (i,j) \in A^k \tag{6}$$

$$T_i^k \geq a_i \ \forall k \in M, \forall i \in N \tag{7}$$

$$T_i^k + t_i^k \sum_{j \in N \cup \{d(k)\}} x_{j,i}^k \leq b_i \ \forall k \in M, \forall i \in N \tag{8}$$

$$T_{o(k)}^k \geq s^k \ \forall k \in M \tag{9}$$

$$T_{d(k)}^k \leq e^k \ \forall k \in M \tag{10}$$

$$x_{i,j}^k \in \{0,1\} \ \forall k \in M, \forall (i,j) \in A^k \tag{11}$$

The objective function minimizes the elapsed time since the ships incoming, mooring and handling, considering a respective service cost. Constraints (2) state that each ship is only once assigned to one berth. Constraints (3) and (4) ensure that a ship will be the first handling by each berth and another will be the last. The flow conservation is given by constraint (5) and constraint (6) calculates the ships berthing time. Only the valid arches A^k ($\forall k \in M$) are considered in constraint (6), because some ships cannot be assisted by certain berth, because

for instance, the type of available equipment in berth cannot be appropriate for handling some load types. The instance's data shown the berth capacity to attend the ships (handling time is different of zero). Constraints (7) and (8) state that the berthing time is posterior to the ship arrival time and completion time happens before the ship's time limit (time window). Constraint (9) and (10) ensure the non time windows violation in berths, and constraint (11) sets that decision variables $x_{i,j}^k$ will be binary.

4 The PTA/LP Method

Initially proposed by Mauri and Lorena [8], the PTA/LP is a heuristic method based on applying the Population Training Algorithm (PTA) and Linear Programming (LP) through the Column Generation technique. The PTA and LP are applied in an interactive way. The PTA uses the information of dual variables in a LP relaxation to generate improved incoming columns (low cost and good covering of the ships) in a column generation process. The LP relaxation is used for solve a Set Partitioning Problem (SPP) formed by these columns. The SPP is formulated as follows:

Minimize:

$$Z^* = \sum_{j=1}^{p} c_j x_j \tag{12}$$

Subject to:

$$\sum_{j=1}^{p} a_{ij} x_j = 1 \;\; i = 1, ..., n \tag{13}$$

$$x_j \in \{0, 1\}; \;\; j = 1, ..., p \tag{14}$$

The BAP is modeled as a matrix constructing with columns representing berths and lines the ships. Each element $a_{ij} \in \{0, 1\}$, $i \in N = 1..n$ and $j \in P = \{1..p\}$. n is the number of ships (lines) and p the number of generated columns. $a_{ij} = 1$ if the column j attends the ship i, and 0 otherwise. This is a classic formulation constantly used in several works found in the literature. The c_j represents the cost of column j (defined in eq. 16) and x_j is equal to 1 if column j belongs to the problem solution and 0 otherwise.

In BAP specific case, each berth has its own features, and sometimes a ship type could not be attended by a berth. Just a berth (or none) of each available "type" in quay must be used (each column belonging to the final problem solution should represent a different berth, without repetitions). Then, a new constraint must be inserted in SPP (eq. 15) to forming a set partitioning problem with an additional constraint (SPP^+).

$$\sum_{j=1}^{p} b_{ij} x_j \leq 1 \;\; i = 1, ..., m \tag{15}$$

Each element $b_{ij} \in \{0,1\}$, $i \in M = \{1..m\}$ and $j \in P = \{1..p\}$. m is the number of available berths, and $b_{ij} = 1$ if the column j represents the berth i.

Now seeing in an evolutionary computation context, each column is represented through an "individual" formed by integers, where the first position indicates the berth referring to a column, and the other positions represent the ships attended by this berth (column). For the columns' cost calculation, the time windows constraints in the BAP model (7-10) are relaxed and moved to objective function considering weight factors (vector $w = [w_0, w_1, w_2]$). This approach is used to facilitate and accelerate the generation of new columns, because the computational cost (time) to generate the columns with the restrictions (7-10) is higher. The columns are evaluated in a relaxed way and its cost will receive a high weight for violations in time windows. The cost of each column (individual) is given by

$$\begin{aligned} c_k = w_0 \sum_{i \in B^k} v_i \left(T_i^k - a_i + t_i^k\right) + \\ w_1 \sum_{i \in B^k} \left(\max\left(0, a_i - T_i^k\right) + \max\left(0, T_i^k + t_i^k - b_i\right)\right) + \\ w_2 \left(\max\left(0, s^k - T_{o(k)}^k\right) + \max\left(0, T_{d(k)}^k + e^k\right)\right) \end{aligned} \tag{16}$$

The generation of all necessary columns to build and solve SPP^+ (eq. 12-15) can be a challenge. So, the LP relaxation is used with PTA to generate a suitable set of columns for commercial solvers (a number of columns to be solved by CPLEX - see [5]). More details on PTA/LP can be seen in [9] and [10].

4.1 The Population Training Algorithm

The Population Training Algorithm - PTA is a kind of evolutionary technique first employed in [11] and derived from the Constructive Genetic Algorithm (CGA) proposed by Lorena and Furtado [12]. The CGA has a number of innovative features compared to traditional genetic algorithms. These include a "ranked" population of dynamic size composed of "schemata" and "structures". The schemata and structures are directly evaluated in a common basis, using a double fitness process, called *fg-fitness*.

The schemata are not used in PTA and the fg-fitness will be performed by heuristics. An individual is considered well adapted if it cannot be better regarding the employed training heuristic. The adaptation in the population training is, therefore, used to guide the search to promising areas.

The two functions used in evolutionary training are defined by *g(k)* = "quality" of column (individual) k (eq. 19), and *f(k)* = *Best* $g(k')|k' \in$ *Neighborhood(k)*. The *f(k)* value is obtained through training heuristic (Fig. 3) and the evolutionary process is developed privileging the individuals presenting small differences $[g(k) - f(k)]$ and small *g(k)*, assigning to them the following *ranks*:

$$\delta(k) = d \times [g_{\max} - g(k)] - [g(k) - f(k)] \tag{17}$$

g_{max} is the cost of worst individual (column) created in initial population and d is a constant percentage of g_{max}. The population is dynamically controlled by an evolution parameter denominated α, and updated as:

$$\alpha = \alpha + Step \times PS \times \frac{\delta_{bst} - \delta_{wst}}{RG} \tag{18}$$

Step is a constant that controls the evolutionary process speed and PS is the current population size. $(\delta_{bst} - \delta_{wst})$ is the variation among the ranks of the best and worst individuals, respectively, and RG is an estimated number of remaining generations to finish the process.

The parameter α is compared to the ranks (eq. 17), and if $\alpha \geq \delta(k)$ the individual k is eliminated from the population. The population at the evolution time α is dynamic in size and can be emptied during the process.

```
1.  CREATE (m empty berths);
2.  CREATE (a list L with all the ships);
3.  ORDER (the list L by ships incoming time);
4.  FOR (each ship j in L, j = 1,2,...,n) DO
5.      SELECT (a berth i, i = 1,2,...,m);
6.      IF (the berth i was unable to handling the ship j)
7.         RETURN (to step 5);
8.      ELSE
9.         ASSIGN (the ship j to berth i);
10.     END-IF;
11. END-FOR;
```

Fig. 1. Distribution heuristic

The initial population is generated through two heuristics: *distribution heuristic* and *programming heuristic*. The distribution heuristic attributes the ships to the berths. This heuristic is based on the distribution heuristic presented by Mauri and Lorena [13] and the FCFS-G heuristic presented by Cordeau et al. [3]. The programming heuristic makes the ships schedule in the berths. The distribution heuristic runs for "initial population size" times.

1. GIVEN (any column k)
2. FOR (each ship i assigned to k) DO
3. $$T_i^k = \begin{cases} \max(a_i, s^k), & i = 1 \\ \max(a_i, T_{i-1}^k + t_{i-1}^k), & i > 1 \end{cases}$$
4. END-FOR;
5. CALCULATE (c_k,g(k),f(k) and δ(k))

Fig. 2. Programming heuristic

The distribution heuristic creates initially m empty berths. The n ships are organized by incoming order on port 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 heuristic ensures that each ship will be assigned to a berth that must be able to attend it. The berthing times may present overlapping and/or time windows violations, for ships or berths. The Figure 1 describes the distribution heuristic.

In programming heuristic the berthing time for each ship and the solution objective function of column k (c_k) are computed. The functions *g(k)* and *f(k)* and rank $\delta(k)$ are also computed. The Figure 2 presents the programming heuristic.

A simple local search heuristic is used as training function *f(k)*, and several alternative individuals (columns) in a neighborhood are evaluated. This heuristic is described in Figure 3.

```
1.  GIVEN (any column k);
2.  k' ← k;
3.  f* = g(k')
4.  FOR (neighborhood size times)
5.      i ← any ship attended by column k';
6.      j ← another ship attended by column k';
7.      CHANGE (the attendance sequence for ships i and j);
8.      EXECUTE (the programming heuristic for column k');
9.      IF (g(k') < f*);
10.         f* ← g(k');
11.     END-IF;
12. END-FOR;
13. f(k) ← f*;
```

Fig. 3. Training heuristic

The used mutation is also based in a local search implemented through a simple change of the handling positions of two ships (randomly selected) assisted by a column (individual). This process is described in Figure 4.

```
1.  GIVEN (any column k);
2.  i ← any ship attended by column k;
3.  j ← another ship attended by column k;
4.  CHANGE (the attendance sequence for ships i and j);
5.  EXECUTE (the programming heuristic for column k);
```

Fig. 4. Mutation

The crossover generates new individuals as follows: two individuals are selected (base and guide) and a new individual is created similar to the base. Each ship assisted by the guide individual is inserted in the new individual if the corresponding berth can attend it. The handling sequence of the new individual is ordered by the ships' arrival time on the port. The crossover is presented in Figure 5.

These operators are enclosed to PTA and its pseudo-code is shown in Figure 6. It is interesting to notice that using these processes the PTA will form populations of several sizes, guided by the objective of selecting low cost columns with an enough covering of the ships. The best columns should include a varied

number of ships. This fact is featured by using the training heuristic that will guide the evolutionary process.

```
1.  GIVEN (a base column k);
2.  GIVEN (a guide column k');
3.  k'' ← clone(k);
4.  FOR (each ship i attended by column k') DO
5.      IF (the berth referring to column k'' was able to attend the ship i)
6.         INSERT (the ship i in column k'');
7.      END-IF;
8.  END-FOR;
9.  ORDER (the attendance sequence for column k'');
10. EXECUTE (the programming heuristic for column k'');
11. INSERT (the column k'' in population);
```

Fig. 5. Crossover

```
1.  CREATE (an initial population);
2.  WHILE (the generation maximum number not be reached)
3.      SELECT (a base individual);
4.      SELECT (a guide individual);
5.      k ← CROSSOVER (base,guide);
6.      IF (rand() < mutation probability) MUTATION (k); END-IF;
7.      CALCULATE (δ(k));
8.      IF (δ(k) > α) INSERT (k in population); END-IF;
9.      ORDER (the population by the individuals rank);
10.     UPDATE (α);
11.     FOR (every k ∈ population) DO
12.        IF (δ(k) ≤ α) ELIMINATE (k); END-IF;
13.     END-FOR;
24. END-WHILE;
```

Fig. 6. PTA algorithm

4.2 PTA and LP interaction

The interaction of PTA with LP is made through the fitness function (function g) of the individuals in PTA. This function is defined using the dual variables of LP. The function g is defined as follows:

$$g(k) = \begin{cases} \frac{c_k}{\sum_{i=1}^{n} \lambda_i a_{ik} + \sum_{i=1}^{m} \lambda_i b_{ik}} \;\; for \;\; \left(\sum_{i=1}^{n} \lambda_i a_{ik} + \sum_{i=1}^{m} \lambda_i b_{ik} > 0 \right) \\ c_k \;\; for \;\; \left(\sum_{i=1}^{n} \lambda_i a_{ik} + \sum_{i=1}^{m} \lambda_i b_{ik} \leq 0 \right) \end{cases} \tag{19}$$

c_k is the cost of column k (eq. 16) and λ_i is the dual variable corresponding to constraint i. Using the concepts of the column generation technique, the reduced

cost of column k (θ_k) inserted in SPP^+ can be calculated through the following equation:

$$\theta_k = c_k - \left(\sum_{i=1}^{n} \lambda_i a_{ik} + \sum_{i=1}^{m} \lambda_i b_{ik} \right) \tag{20}$$

We can observe through equations (19) and (20) that for negative costs ($\theta_k < 0$) the value of function g will be situated inside of the interval $[0, 1]$. Therefore, the training heuristic that defines the corresponding function f values (best g in a neighborhood) will assign small differences (g - f) for columns that have negative reduced costs. For positive costs ($\theta_k \geq 0$) the value of the g function will be the respective cost (a "high" value). So, the population is indirectly trained for individuals with negative reduced costs, improving the ship's covering for SPP^+, avoiding the generation of an excessive number of columns and consequently speeding up the process of column generation. The Figure 7 presents the pseudo-code of PTA/LP.

```
1.  CREATE (an initial set of columns);
2.  SOLVE (LP);
3.  FOR (iterations number or maximum number of columns are not reached)
4.      EXECUTE (PTA);
5.      REMOVE (invalid columns);
6.      CALCULATE (reduced cost for new columns);
7.      ADD (columns with negative reduced cost);
8.      SOLVE (LP);
9.  END-FOR;
10. CONVERT (LP for ILP);
11. SOLVE (ILP);
```

Fig. 7. PTA/LP algorithm

In PTA/LP, an initial set of columns addressed to the problem is randomly created. This set must contain columns that form a feasible solution for SPP^+. These columns are generated running the distribution heuristic (Fig. 1) followed by the programming heuristic (Fig. 2) for each column. The solution formed by these columns will be probably invalid, because the columns can present time windows violations. However, the columns with high costs (due to the weights) will be removed from new SPP^+ solutions when improved columns were generated by PTA.

A SPP^+ is formed by the initial set of columns and the LP relaxation is solved by CPLEX. New columns are generated through PTA considering the values of the dual variables to build the fitness functions. The valid columns (that do not present violations in time windows) that present negative reduced costs are added to current SPP^+ and it is solved again through the LP relaxation. These processes are repeated by a certain number of iterations or while a maximum number of generated columns is not reached.

The final SPP^+ is converted to an integer linear problem and solved by CPLEX (through the CPLEX callable library - see [5]). A feasible solution for SPP^+ is obtained, and this solution should be valid and closed of optimal for the BAP model (eq. 1-11).

5 Computational Experience

Several experiments were performed over 30 different instances (60 ships and 13 berths). These instances were randomly generated by Cordeau et al. [3]. All the computational tests were accomplished in a PC with *AMD Athlon* 64 3500 of 2.2 GHz processor with 1GB of RAM and the code was implemented in C++.

Table 1. PTA/LP details

Instance name	Number of generated columns	SPP^+ solved by LP	SPP^+ solved by ILP	Processing time (s) PTA/LP	ILP	Total
i01	26664	1409.00	1409	72.14	2.47	74.61
i02	12752	1261.00	1261	58.92	1.83	60.75
i03	70000	1129.00	1129	94.62	40.83	135.45
i04	54612	1302.00	1302	103.16	7.02	110.17
i05	70019	1207.00	1207	72.20	52.50	124.70
i06	25990	1261.00	1261	74.22	4.12	78.34
i07	70023	1279.00	1279	86.73	27.47	114.20
i08	70005	1299.00	1299	48.77	8.30	57.06
i09	37846	1444.00	1444	91.86	4.61	96.47
i10	70005	1213.00	1213	61.81	37.59	99.41
i11	43507	1369.00	1369	95.34	4.00	99.34
i12	18508	1325.00	1325	77.39	3.30	80.69
i13	70017	1360.00	1360	62.55	27.39	89.94
i14	26221	1233.00	1233	69.05	4.91	73.95
i15	70002	1295.00	1295	71.28	2.91	74.19
i16	30063	1365.00	1365	169.81	0.55	170.36
i17	70033	1283.00	1283	32.89	13.67	46.58
i18	36108	1345.00	1345	81.78	2.23	84.02
i19	16135	1367.00	1367	122.00	1.19	123.19
i20	20528	1328.00	1328	74.25	8.05	82.30
i21	48386	1341.00	1341	103.52	4.56	108.08
i22	54140	1326.00	1326	104.17	1.20	105.38
i23	70010	1266.00	1266	41.59	2.12	43.72
i24	70008	1260.00	1260	75.81	3.09	78.91
i25	41210	1376.00	1376	95.09	1.48	96.58
i26	70011	1318.00	1318	70.00	31.11	101.11
i27	37022	1261.00	1261	77.38	5.48	82.86
i28	70004	1360.00	1360	51.52	1.39	52.91
i29	70001	1280.00	1280	196.36	7.00	203.36
i30	7837	1344.00	1344	69.62	1.39	71.02
Average	48256	1306.87	1306.87	83.53	10.46	93.99

The control parameters used by PTA/LP are presented as follows. The initial population size was set to 10; the *Step* parameter was set to 0.001; the maximum number of generations was 70; the base percentage and the mutation probability was set to 10 and 60 respectively; the neighborhood size was set to 6, and the parameter d was set to 0.01; the maximum number of columns was limited to 70000, and the iterations number was set to 10000. In all of the experiments the values of g_{max} were obtained from the largest g evaluation on individuals

generated in the initial population. The initial value of α was set to 0 and the weights were set to w = [1,10,10].

The Table 1 presents some details of the PTA/LP performance. The solution value of the last SPP^+ (formed by all the generated columns) was the same when solved by LP and ILP. This fact indicates that optimal solutions are found for the SPP^+ formed by the generated columns subset (these solutions should be close of the original problems optimal). The interaction time for PTA and LP and the time for final SPP^+ resolution through ILP were relatively low resulting in a competitive total time of processing for PTA/LP.

In Table 2 the column "A" presents the improvement obtained by PTA/LP over Tabu Search (TS). The column "B" presents the improvement of PTA/LP over CPLEX. The solutions obtained by PTA/LP were compared against the best known solutions for the used instances. These best solutions were obtained through a Tabu Search heuristic presented in [3]. Besides, the CPLEX 10.0.1 [5] was also used in an isolated way to solve the model described in Section 3. The CPLEX was unable to find solutions for several instances (see Table 2). The CPLEX and Tabu Search, respectively, spent 1 hour (3600 seconds) and approximately 120 seconds of processing time for solving each instance [3], while PTA/LP spent an average of 93.99 seconds for each instance. This fact shows the PTA/LP competitiveness over Tabu Search and CPLEX.

Table 2. Comparison against other methods

Instance	TS	CPLEX		PTA/LP	Improvements (%)	
name	Z	Z	Gap	Z*	A	B
i01	1415	-	-	1409	0.43	-
i02	1263	2606	3.82	1261	0.16	106.66
i03	1139	2565	4.00	1129	0.89	127.19
i04	1303	4353	8.62	1302	0.08	234.33
i05	1208	2672	4.89	1207	0.08	121.38
i06	1262	-	-	1261	0.08	-
i07	1279	2887	4.73	1279	0.00	125.72
i08	1299	5177	11.69	1299	0.00	298.54
i09	1444	-	-	1444	0.00	-
i10	1212	-	-	1213	-0.08	-
i11	1378	-	-	1369	0.66	-
i12	1325	3206	5.48	1325	0.00	141.96
i13	1360	-	-	1360	0.00	-
i14	1233	-	-	1233	0.00	-
i15	1295	4672	9.77	1295	0.00	260.77
i16	1375	4320	8.97	1365	0.73	216.48
i17	1283	-	-	1283	0.00	-
i18	1346	3681	6.94	1345	0.07	173.68
i19	1370	2400	3.04	1367	0.22	75.57
i20	1328	-	-	1328	0.00	-
i21	1346	-	-	1341	0.37	-
i22	1332	3489	7.31	1326	0.45	163.12
i23	1266	-	-	1266	0.00	-
i24	1261	4867	10.13	1260	0.08	286.27
i25	1379	1993	2.67	1376	0.22	44.84
i26	1330	2520	3.62	1318	0.91	91.20
i27	1261	3209	5.70	1261	0.00	154.48
i28	1365	-	-	1360	0.37	-
i29	1282	4809	9.43	1280	0.16	275.70
i30	1351	-	-	1344	0.52	-
Average	1309.67	3495.65	6.52	1306.87	0.21	170.46

6 Conclusions

This work presented a new hybrid column generation technique to solve the BAP. The PTA integrated with a traditional column generation technique solves column generation sub-problems in an implicit way. The definition of the PTA *fg-fitness* using dual variables information is the essential feature for PTA/LP performance. The computational results were very good and obtained in reasonable processing times compared against the Tabu Search and CPLEX.

The proposed approach doesn't guarantee to find of optimal solutions for BAP, because the column generation sub-problem was solved through a heuristic method. However, the results show good quality solutions, which are probably close to the optimal, suggesting the application to real problems of Brazilian ports and other similar problems.

References

1. Imai, A., Nishimura, E., Papadimitriou, S.: Berthing ships at a multi-user container terminal with a limited quay capacity. Transportation Research - Part E (2006)
2. Vis, I.F.A., Koster, R.D.: Transshipment of containers at a container terminal: An overview. European Journal of Operational Research **147** (2003) 1–16
3. Cordeau, J.F., Laporte, G., Legato, P., Moccia, L.: Models and tabu search heuristics for the berth allocation problem. Transportation Science **39** (2005) 526–538
4. Filho, G.R., Lorena, L.A.N.: Constructive genetic algorithm and column generation: an application to graph coloring. In Proceedings of APORS 2000 - The Fifth Conference of the Association of Asian-Pacific Operations Research Societies within IFORS (2000)
5. ILOG France: ILOG CPLEX 10.0 - User's Manual. (2006)
6. Puchinger, J., Raidl, G.R.: Models and algorithms for three-stage two-dimensional bin packing. European Journal of Operational Research. Feature Issue on Cutting and Packing (2006)
7. Cordeau, J.F., Laporte, G. Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society **52** (2001) 928–936
8. Mauri, G.R., Lorena, L.A.N.: Método interativo para resolução do problema de escalonamento de tripulações. XXXVI Brazilian Symposium of Operational Research (2004)
9. Mauri, G.R.: Novas heurísticas para o problema de escalonamento de tripulações. Master Thesis in Applied Computing. Brasilian Institute for Space Research (2005)
10. Mauri, G.R., Lorena, L.A.N.: A new hybrid heuristic for driver scheduling. International Journal of Hybrid Intelligent Systems **1**(4) (2007) 39–47
11. Oliveira, A.C.M., Lorena, L.A.N.: 2-opt population training for minimization of open stack problem. In: Bittencourt G, Ramalho GL (Eds), Advances in Artificial Intelligence, Springer Lecture Notes in Artificial Intelligence Series **2507** (2002) 313–323
12. Lorena, L.A.N., Furtado, J.C.: Constructive genetic algorithm for clustering problems. Evolutionary Computation **3**(9) (2001) 309–327
13. Mauri, G.R., Lorena, L.A.N.: Simulated annealing aplicado a um modelo geral do problema de roteirização e programação de veículos. XXXVIII Brazilian Symposium of Operational Research (2006)