Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

# A Grasp+Vnd Algorithm for a Class of Job Scheduling Problem in Parallel Machines

**Dalessandro Soares Vianna**[a]
[a]Fluminense Federal University (UFF), Rio das Ostras, Brazil

**Sandra Regina Coelho**[b]
[b]Candido Mendes University (UCAM), Campo, Brazil

**Marcilene de Fátima Dianin Vianna**[c]
[c]Fluminense Federal University (UFF), Campos, Brazil

## Abstract

Petróleo Brasileiro S/A (Petrobras) is the biggest Brazilian company in energy business. It acts in exploration, production, refinement, commercialization and transport of petroleum by products in Brazil and other countries. Most of the petroleum production is concentrated in the Campos basin, where the company port (port of Imbetiba – Macaé/RJ) is located. All the oil rigs supply is done using this port. Researches made at the port of Imbetiba show the need of optimizing, among others, the problem of towboat scheduling. In this problem, the order of towboat attendance must be decided and, according to the material that each one carry, the following restrictions must be respected: each towboat has a subset of piers where it can be attended; each one has a attendance priority; and each one has a minimal time where it can be attended, that is, before this time it cannot be attended. This paper proposes a GRASP algorithm for the problem of towboat scheduling, which utilizes the technique VND as local search. Three neighborhood structures are used: Exchange, Interchange and Relocation. Computational results show that the proposed algorithm is efficient when compared with traditional GRASP algorithms.

***Keywords:*** *Job Scheduling Problem in Parallel Machines, GRASP, VND, Combinatorial Optimization, Petrobras.*

## Introduction

Petróleo Brasileiro S/A (PETROBRAS) commenced its activities in the 20th century and it is nowadays the biggest Brazilian company in energy business. It acts in exploration, production, refinement, commercialization and transportation of petroleum by products in Brazil and other countries. The organizational structure regulates the operation establishing four business areas: exploration & production,

BJ Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

supply, gas & energy and international, and two of support: financial and services, besides the corporative units connected straightly to the president (PETROBRAS, 2008).

The company has 112 oil rigs, being 78 fixed and 34 floating, 16 refinery, 25,197 kilometers of ducts and 6,154 gas station spread on the national territory. Among these ones, 631 are from their own (PETROBRAS, 2008).

Most of the petroleum production of the company is concentrated in Campos basin, where is located the own port of the company (port of Imbetiba – Macaé/RJ). Through this port it is done all the supply of the oil rigs of the basin.

Researches made at the port of Imbetiba show the need of optimizing, among others, the problem of towboat scheduling. In this problem, the order of attendance must be decided and, according to the material each towboat carries, the following restrictions must be respected:

- Each towboat has a subset of piers that can attend it. This happens because each pier is prepared to the embark / disembark of a set of materials;
- Each towboat has a priority of attendance. The priority is given according to the urgency of the material, the type of the material (perishable or not), or taking priority to the attendance to an oil rig that in that moment is with a higher level of production, among other factors;
- Minimum time for attendance (release time), that represents the moment that the oil rig is already at disposal.

The objective is to minimize the waiting time of the towboats in the port, what can represent very high amounts for PETROBRAS and for the contracted companies that use the port of Imbetiba.

This problem can be associated to the Job Scheduling Problem in Parallel Machines – JSPPM (MENDES *et al*., 2002; ARROYO and RIBEIRO, 2004; ZOUBA, BAPTISTE and REBAINE, 2009), where a set of jobs, each one with a execution time, must be organized for execution in a set of identical machines, in a way that the total time of execution should be minimized. In the studied problem, the jobs are the towboats and the machines are the piers, that are not identical – each one prepared to attend the demand of a subset of materials. Besides, the restriction of priority and of release time must be considered. The existence of this set of restrictions is because the port of Imbetiba is a private and small port, having own rules different from the ones established in big ports. This fact turned impossible to compare the research made in this paper with others already made for other ports.

This paper proposes a GRASP algorithm that uses the VND technique in the local search phase. Three neighborhood structures are implemented: Exchange, Interchange and Relocation. The proposed algorithm is compared to traditional GRASP algorithms. Computational results show that the proposed algorithm is more efficient than the other algorithms when the quality of the solution obtained is compared and, also, when the computational times are compared.

Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

The remaining of this paper is organized as following: in Section 2, the JSPPM, the related literature and the class of JSPPM cited in this paper are presented. The GRASP algorithm proposed, which uses VND technique in the local search phase, is detailed in Section 3. In Section 4, the computational results are presented. Finally, the conclusion and references are presented.

## Job Scheduling Problem in Parallel Machines

According to Mendes *et al.* (2002), the JSPPM can be defined as following: a set of *n* jobs must be organized in *m* identical parallel machines with the aim of minimizing the makespan – enough time to the *n* jobs complete its execution. Each machine has a dependent setup time, i.e., the machine changeover time is dependent on which job is currently being processed on the machine.

In the literature related to JSPPM, it is found: a integer linear programming model, proposed by Dearing and Henderson (1984), for loom assignment in a textile weaving operation; an heuristic method, proposed by Sumichrast and Baker (1987), based on the solution of a series of 0-1 integer subproblems that improves the results of Dearing and Anderson (1984); tabu search heuristics are proposed by França *et al.*(1996) and Çelik and Saricicek (2009); Arroyo and Ribeiro (2004) proposed a genetic algorithm for the JSPPM with multiple objectives; a tabu search heuristic, proposed by Mendes *et al.* (2002) for the JSPPM not preemptive; and a genetic algorithm is proposed by Zouba, Baptiste and Rebaine (2009).

The class of problem related in this paper, JSPPM*, can be defined as following: a group of *n* jobs must be organized in *m* machines not identical. The setup time is independent, in other words, it is the same independent of whom was the job that has just been executed. So, the setup times are not considered in this paper. Each job has a subset of machines that can execute it, a minimum time for its attendance (release time) and a priority.

Each job *i* has a priority $p_i$. As bigger is the value of $p_i$, more priority the job *i* has. Each job *i* also has a release time $rt_i$. Being $at_i$ the moment when the job *i* started effectively its attendance, the waiting time, $te_i$, of the job *i*, can be calculated as following: $te_i = at_i - rt_i$. The goal of the JSPPM* is to minimize the total waiting time, considering the priority of each job. The total waiting time can be calculated as following:

$$\sum_{i=1}^{n} p_i \times te_i = \sum_{i=1}^{n} p_i \times (at_i - rt_i)$$

## Grasp + Vnd Heuristic Proposed

GRASP – Greedy Randomized Adaptive Search Procedure – (FEO and RESENDE, 1995; RESENDE and RIBEIRO, 2003) is a multi-start metaheuristic, in which each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution using a greedy randomized algorithm,

BJ Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

while the local search phase calculates a local optimum in the neighborhood of the feasible solution. Both phases are repeated a pre-specified number of iterations and the best overall solution is kept as the result.

In this paper is proposed an hybrid GRASP algorithm that uses the VND – *Variable Neighborhood Descent* – (MLADENOVIC and HANSEN, 1997) technique in the local search phase. The construction method is described in details in Subsection 3.1. In the local search phase is used the VND technique with three neighborhood structures that will be commented in Subsection 3.2.

***Constructive Method***

This method is greedy according to the number of machines that can execute a specific job. Jobs with a smaller number of machines that can execute them will be selected first. The main goal of this strategy is to avoid that a job *i* waits for a machine that is used by job *j* that could be attended by other machines that are idle.

In this way, initially, the jobs are sorted increasingly by the number of machines that can attend them. When more than one job has the same quantity of machines to attend it, the decision is given by the priority of the jobs (taking into consideration the decreasing order of the priorities).

In order to construct an different initial solution at each iteration of the GRASP algorithm, instead of always choosing the first job of the queue, it is chosen, randomly, one among the first *a* jobs, where *a* is an input parameter of the algorithm.

To associate a job *i* with a machine $m_j$, the best position must be chosen considering all the available positions in the machine $m_j$, in a way that the total waiting time be the smallest possible. When a job can be executed by more than one machine, it will be chosen the machine where the job does not cause waste of time or cause the smallest possible waiting time.

Figure 1 shows the constructive algorithm, that receives as input parameter the randomness intensity, *a*, and returns, as output, the solution *s* built. In line 1, the list with all the jobs is ordered increasingly by the amount of machines that can attend each job and by the priority. The loop in lines 2-16 guarantees that all the jobs will be inserted in the solution *s*. In line 3, a job *b* is chosen randomly among the first *a* jobs of the list. The loop in lines 5-13 chooses, in the machines that can attend the job *b*, the best position to insert the job *b*. This insertion is done in line 14. In line 17, the solution *s* is returned.

Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

**Procedure BuildSolution** ( *a* )

**Input**

*a* – randomness intensity.

**Output**

*s* – built solution.

***Begin***

01. Construct the list *L* with jobs not attended yet. Sort *L* increasingly by the amount of machines that can attend each job. In case of equality, sort decreasingly by the priority of the job;

**02. For** $i \leftarrow 1$ **to** *n* **do**

**03.** $b \leftarrow$ choose, randomly, one of the first *a* jobs of *L*;

04. $best\_cost \leftarrow \infty$ ;

05. **For** each machine $m_j$ that can attend the job *b* **do**

06. Let *y* be the best position in the machine $m_j$ to insert the job *b*, that is, the position that causes the smallest waiting time;

*07.* $cost\_y \leftarrow$ cost of the insertion of job *b* in the position *y* of the machine $m_j$;

08. **If** $cost\_y < best\_cost$ **then**

09. $best\_pos \leftarrow y$;

10. $best\_mac \leftarrow m_j$;

11. $best\_cost \leftarrow cost\_y$;

12. **End-if**

**13. End-for**

**14.** Insert the job *b* in the position *best_pos* of the machine *best_mac* of solution *s*.

Remove *b* from *L*;

15. Calculate the new accumulated waiting time of the machine *best_mac*;

**16. End-for**

**17. Return** *s;*

**End-BuildSolution**

Figure 1 - Constructive algorithm.

BJ Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

***Local Search Using VND***

Figure 2 shows the pseudo-code of the VND algorithm developed that uses three neighborhood structures: *Exchange, Interchange and Relocation,* that will be described in details in Subsections 3.2.1, 3.2.2 and 3.2.3, respectively. This algorithm receives as input parameter, besides the three neighborhood structures, a feasible solution $s_0$ to be refined. The first neighborhood structure to be analyzed is the neighborhood $N_1$ (line 2). The loop in lines 3-11 guarantees that all the neighborhoods structures will be analyzed. If, after the application of the local search in the neighborhood $N_k$, the found solution is better than the original, the search will restart (line 7) with the neighborhood $N_1$. Otherwise, the search will continue in the next neighborhood (line 9). In line 12, the refined solution *s* is returned.

*Exchange Neighborhood Structure*

In this neighborhood structure, it is considered the exchange of jobs that are in the attendance queue of the same machine. Figure 3 shows the pseudo-code of the local search procedure (**LS_Exchange**) analyzing this neighborhood structure, which receives as input parameter the solution *s* to be refined. The loop in lines 2-20 guarantees that the search will continue while exists a better neighbor than the current solution *s*. The loop in lines 4-13 searches for the best exchange among jobs of the same machine. If the best exchange takes to a better solution than the current one, this exchange will be done in line 15. Otherwise, the procedure is concluded. The solution refined *s* is returned in line 21.

*Interchange Neighborhood Structure*

In this neighborhood structure, it is considered the exchange of jobs that are in the attendance queue of different machines. Figure 4 shows the pseudo-code of the local search procedure (**LS_Interchange**) using this neighborhood structure, which receives as input parameter the solution *s* to be refined. The loop in lines 2-21 guarantees that the search will continue while exists a better neighbor than the current solution *s*. The loop in lines 4-14 searches for the best exchange among jobs of different machines. If the best exchange takes to a better solution than the current one, this exchange will be done in line 16. Otherwise, the procedure is concluded. The solution refined *s* is returned in line 22.

*Relocation Neighborhood Structure*

In this neighborhood structure, it is considered the migration of a job *i* from the current machine to another one that can also attend the job *i*. Figure 5 shows the pseudo-code of the local search procedure (**LS_Relocation**) using this neighborhood structure, which receives as input parameter the solution *s* to be refined. The loop in lines 2-22 guarantees that the search will continue while exists a better neighbor than the current solution *s*. The loop in lines 4-16 searches for the best relocation that can be done. If the best relocation takes to a better solution than the current one, this relocation will be done in line 18. Otherwise, the procedure is concluded. The solution refined *s* is returned in line 23.

Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

**Procedure VND** ( $s_0$, $N_1$, $N_2$, $N_3$ )

**Input**

$s_0$ – a feasible solution;

$N_1$ – *Exchange* neighborhood structure;

$N_2$ – *Interchange* neighborhood structure;

$N_3$ – *Relocation* neighborhood structure.

**Output**

$s$ – refined solution.

**Begin**

01. $s \leftarrow s_{0;}$
02. $k \leftarrow 1$;
03. **While** $k \leq 3$ **do**
04. Apply the local search procedure in $s$ using the neighborhood structure $N_{k.}$ Let $s'$ be the local optimum;
05. **If** $f(s') < f(s)$ **then**
06. $s \leftarrow s'$;
07. $k \leftarrow 1$;
08. **Else**
09. $k \leftarrow k+1$;
10. **End-if**
11. **End-while**
12. **Return** $s$;

**End-VND**

Figure 2 - VND procedure used.

Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

**Procedure LS_Exchange** ( *s* )

**Input**

*s* – feasible solution.

**Output**

*s* – refined solution

**Begin**

01. *Finished* ← **false**;
02. **While not** *finished* **do**
03. *Best_change* ← ∞;
04. **For** each job *i* **do**
05. **for** each job *j* (*j* ≠ *i*) of the same machine of *i* **do**
06. *c* ← additional cost caused by the exchange of jobs *i* and *j*;
07. **If** *c* < *Best_change* **then**
08. *Best_change* ← *c*;
09. *Best_i* ← *i*;
10. *Best_j* ← *j*;
11. **End-if**
12. **End-for**
13. **End-for**
14. **If** *Best_change* < 0 **then**
15. Make the exchange of jobs *Best_i* and *Best_j*;
16. Evaluate the solution *s* after the exchange;
17. **Else**
18. *finished* ← **true**;
19. **End-if**
20. **End-while**
21. **Return** *s*;

**End- LS_Exchange**

Figure 3 - LS_Exchange algorithm.

Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

**Procedure LS_Interchange** ( *s* )

**Input**

*s* – feasible solution.

**Output**

*s* – refined solution

**Begin**

01. *Finished* ← **false**;
02. **While not** *finished* **do**
03. *Best_change* ← ∞;
04. **For** each job *i* **do**
05. *Machine_i* machine that attends the job *i*;
06. **for** each job *j* of the machine *Machine_j* (*Machine_i* ≠ *Machine_j*) that can attend the job *i* **do**
07. *c* ← additional cost caused by the exchange of jobs *i* and *j*;
08. **If** *c* < *Best_change* **then**
09. *Best_change* ← *c*;
10. *Best_i* ← *i*;
11. *Best_j* ← *j*;
12. **End-if**
13. **End-for**
14. **End-for**
15. **If** *Best_change* < 0 **then**
16. Make the exchange of jobs *Best_i* and *Best_j*;
17. Evaluate the solution *s* after the exchange;
18. **Else**
19. *finished* ← **true**;
20. **End-if**
21. **End-while**
22. **Return** *s*;

**End- LS_Interchange**

Figure 4 - LS_Interchange algorithm.

BJ Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

**Procedure LS_Relocation** ( *s* )

**Input**

*s* – feasible solution.

**Output**

*s* – refined solution

**Begin**

01. *Finished* ← **false**;
02. **While not** *finished* **do**
03. *Best_change* ← ∞;
04. **For** each job *i* **do**
05. *Machine_i* ← machine that attends the job *i*;
06. **for** each position *pos* of the machine *Machine_j* (*Machine_i* ≠ *Machine_j*) that can attend the job *i* **do**
07. *c* ← additional cost caused by the relocation of job *i* to the position *pos* of *machine_j*;
08. **If** *c* < *Best_change* **then**
09. *Best_change* ← *c*;
11. *Best_i* ← *i*;
12. *Best_machine* ← *machine_j*;
13. *Best_pos* ← *pos*;
14. **End-if**
15. **End-for**
16. **End-for**
17. **If** *Best_change* < 0 **then**
18. Make the relocation of job *Best_i* to the position *Best_pos* of *Best_machine*;
19. Evaluate the solution *s* after the relocation;
20. **Else**
21. *finished* ← **true**;
22. **End-while**
23. **Return** *s*;

**End- LS_ Relocation**

Figure 5 - LS_Relocation algorithm.

Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

## Computational Results

All the computational experiments of this paper were done in a R300 Presario Compaq Notebook with 3200 Athlon Processor and 520 Mb of RAM memory.

The proposed algorithm, as well as the GRASP algorithms used as comparison, was implemented using the C programming language.

The GRASP algorithms that were implemented for validation of the proposed algorithm will be presented in Subsection 4.1. The test problems used in the experiments will be discussed in Subsection 4.2. The experiments done will be shown in Subsection 4.3.

### *GRASP Heuristics*

It is implement three traditional GRASP algorithms. All of them use the constructive algorithm described at Subsection 3.1 and respect the standard GRASP procedure shown in Figure 6. What makes the difference among these strategies is the structure of the neighborhood used in the local search phase.

The three implemented GRASP algorithms are:

- **GRASP_Ex** – uses the procedure **LS_Exchange** in the local search phase;
- **GRASP2_Inter** – uses the procedure **LS_Interchange** in the local search phase;
- **GRASP2_Re** – uses the procedure **LS_Relocation** in the local search phase.

### *Test problems*

For making a comparison among the proposed algorithm and the GRASP algorithms, described in Subsection 4.1, it is used 16 test problems, created in this paper, which uses parameters that are close to a real case of the problem of optimizing the waiting time of the towboats in the port of Imbetiba (Macaé/RJ), problem that motivated this paper. These test problems have $m$ = 5 to 8 piers (machines) and the time space to be scheduled varying from $T$ = 48 to 120 hours. Table 1 shows, for each test problem, the number of piers ($m$) and the total time for scheduling ($T$).

BJ Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

**Procedure GRASP** ( *N_iter*, *a* )

**Input**

*N_iter* – number of GRASP iterations;

*a* - defines the randomness intensity of the algorithm.

**Output**

*Sol* – best solution found.

**Begin**

01. **For** *i* ← 1 **to** *N_iter* **do**
03. *s* ← **Construtive_Algorithm** (*a*);
04. s'← **LocalSearch_Algorithm** (*s*);
05. **If** *s*' is the best solution until this moment **then**
06. *Sol* ← *s*';
07. **End-if**
08. **End-for**
09. **Return** *Sol*;

**End_GRASP**

Figure 6 - Standard GRASP algorithm.

Table 1 - Test Problems.

| Test problem | Number of piers (*m*) | Time (*T*) in hours |
|---|---|---|
| Ins5-48 | 5 | 48 |
| Ins5-72 | 5 | 72 |
| Ins5-96 | 5 | 96 |
| Ins5-120 | 5 | 120 |
| Ins6-48 | 6 | 48 |
| Ins6-72 | 6 | 72 |
| Ins6-96 | 6 | 96 |
| Ins6-120 | 6 | 120 |
| Ins7-48 | 7 | 48 |
| Ins7-72 | 7 | 72 |
| Ins7-96 | 7 | 96 |
| Ins7-120 | 7 | 120 |
| Ins8-48 | 8 | 48 |
| Ins8-72 | 8 | 72 |
| Ins8-96 | 8 | 96 |
| Ins8-120 | 8 | 120 |

Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

The other parameters of these test problems were defined as following: the *release time* (*rt*) of each towboat was defined randomly between 0 and (*T*-1) hours. The time spent on embark/disembark of a towboat was defined randomly between 1 and 24 hours. The amount of towboats, *n*, of the problem was defined by the following formula: (*m*T*)/12, where the denominator represents the medium time of attendance (embark/disembark) of a towboat (job). The priority of attendance of each towboat was defined randomly in the interval [0; 5].

***Experiments Done***

In the first experiment done, the proposed **GRASP+VND** algorithm and each GRASP algorithm, described in Subsection 4.1, were executed during *N_iter*=1000 iterations for each test problem described in Table 1. This procedure was repeated five times, varying the seed of generation of random numbers. In this paper, a version in the C programming language of the generator described in (SCHRAGE, 1979) was used for the generation of the random numbers.

Table 2 - Results of the first experiment.

| *Test problem* | **GRASP_Ex** | | **GRASP_Inter** | | **GRASP_Re** | | **GRASP+VND** | |
|---|---|---|---|---|---|---|---|---|
| | *Cost* | *Time* | *Cost* | *Time* | *Cost* | *Time* | *Cost* | *Time* |
| Ins5-48 | 332 | 61 | 335 | 69,9 | **323** | 72,5 | 327 | 15 |
| Ins5-72 | 571 | 85,4 | 607 | 102,7 | 571 | 108,8 | **568** | 22,76 |
| Ins5-96 | 1022 | 129,5 | 1052 | 200,6 | 1085 | 208,3 | **992** | 44,12 |
| Ins5-120 | 1536 | 261,8 | 1585 | 366,4 | 1448 | 388,7 | **1423** | 80 |
| Ins6-48 | 425 | 96,2 | 426 | 113,1 | **414** | 116,6 | 421 | 24,44 |
| Ins6-72 | 1169 | 140,2 | 1169 | 176,8 | 1159 | 188,1 | **1147** | 39,02 |
| Ins6-96 | 1650 | 280,1 | 1693 | 379,6 | 1623 | 398,8 | **1620** | 82,14 |
| Ins6-120 | 1054 | 432,8 | 1006 | 578,3 | 1050 | 613,3 | **958** | 125,5 |
| Ins7-48 | 573 | 128,3 | 561 | 149 | 561 | 153,9 | **560** | 32,66 |
| Ins7-72 | 1194 | 212,8 | 1516 | 267,9 | 1144 | 284,1 | **1136** | 59,08 |
| Ins7-96 | 2161 | 333,1 | 2142 | 522,5 | **1901** | 551,8 | 1968 | 115,5 |
| Ins7-120 | 2058 | 640,7 | 2138 | 898,9 | **1870** | 969,4 | 1987 | 196,34 |
| Ins8-48 | 778 | 167,2 | 788 | 186,2 | 784 | 194,2 | **767** | 40,8 |
| Ins8-72 | 1830 | 336,5 | 1791 | 427,3 | 1796 | 447,2 | **1770** | 94,78 |
| Ins8-96 | 1656 | 561,4 | 1727 | 751,5 | 1731 | 797,5 | **1657** | 163,8 |
| Ins8-120 | 1970 | 824,2 | 1979 | 1187 | 1943 | 1267,7 | **1822** | 259,36 |

Table 2 shows the costs (total waiting times) and the times (in seconds) obtained, for each test problem, by the proposed algorithm, **GRASP+VND**, and by the other GRASP algorithms. In this table, the best costs obtained for each test problem were emphasized in boldface type. The **GRASP+VND** algorithm got the best costs for 12 among the 16 test problems. For the other 4, the **GRASP_Re** algorithm that uses the *Relocation* neighborhood structure got the best costs. When the computational times were compared, the **GRASP+VND** algorithm always got better times. This happens because, using the different structures of neighborhood, the convergence in the local search phase is faster.

BJ Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

Table 3 - Results of the second experiment.

| *Test problem* | **GRASP_Ex** *Cost* | **GRASP_Inter** *Cost* | **GRASP_Re** *Cost* | **GRASP+VND** *Cost* |
|---|---|---|---|---|
| Ins5-48 | 346 | 337 | 370 | **327** |
| Ins5-72 | 571 | 579 | 571 | **568** |
| Ins5-96 | 1023 | 1067 | 1084 | **992** |
| Ins5-120 | 1538 | 1549 | 1599 | **1423** |
| Ins6-48 | 440 | 426 | 456 | **421** |
| Ins6-72 | 1159 | 1167 | 1169 | **1147** |
| Ins6-96 | 1646 | 1673 | 1689 | **1620** |
| Ins6-120 | 1044 | 1052 | 1037 | **958** |
| Ins7-48 | 567 | 561 | 573 | **560** |
| Ins7-72 | 1224 | 1577 | 1592 | **1136** |
| Ins7-96 | 2052 | 2012 | **1962** | 1968 |
| Ins7-120 | 2242 | 2179 | 2450 | **1987** |
| Ins8-48 | 754 | 790 | 784 | **767** |
| Ins8-72 | 1783 | 1779 | 1847 | **1770** |
| Ins8-96 | 1713 | 1710 | 1718 | **1657** |
| Ins8-120 | 1990 | 1943 | 2066 | **1822** |

Once the **GRASP+VND** algorithm always showed the smallest time in the first experiment, it was done a second experiment, where it was checked how the other algorithms would behave in case their execution would be interrupted when they got the computational time spent by the **GRASP+VND** algorithm. This way, the stop criterion of the GRASP algorithms was changed to the time spent by the **GRASP+VND** algorithm. Table 3 shows the costs obtained by each algorithm.

In this new experiment, it can be seen the superiority of the **GRASP+VND** algorithm. In 15 of the 16 test problems, the cost obtained by the proposed algorithm was better than the other algorithms. Only in the instance "Ins7-96" the proposed algorithm was outperformed per 0.3%.

## Conclusion

This paper came from the necessity of solving the problem of minimizing the waiting time of towboats in the port of Imbetiba (Macaé/RJ). This port, because of being a small one and exclusive from PETROBRAS, has its own rules that turn impossible the comparison to other researches done to other ports. This problem was associated to the problem of job scheduling in parallel machines (ARROYO and RIBEIRO, 2004; DEARING and HENDERSON, 1984; FRANÇA *et al.*, 1996; MENDES *et al.*, 2002; SUMICHRAST and BAKER, 1987; ZOUBA, BAPTISTE, and REBAINE, 2009), which is NP-difficult. This way, a good strategy to solve it is the use of metaheuristics (DRUMMOND *et al.*, 2001; OCHI *et al.*, 1998; VIANNA *et al.*, 1999).

The hybrid GRASP algorithm proposed, **GRASP+VND**, was, in the experiments done, efficient to the class of job scheduling problem in parallel machines cited in this paper. In one of the experiments done, where the same time of execution

Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

was given to all the algorithms, the proposed algorithm got the best results for 15 of the 16 test problems, being outperformed only in one instance by a small difference. This shows that using a procedure that utilizes different neighborhood structures, like the VND, can bring benefits when incorporated to the metaheuristic GRASP.

## Acknowledgements

This work was finantiated by: Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq); Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ); Parque de Alta Tecnologia do Norte Fluminense (TECNORTE); and Fundação Estadual do Norte Fluminense (FENORTE).

## References

Arroyo, J. E. C. and Ribeiro, R. L. P. (2004), "Algoritmo Genético para o Problema de Escalonamento de Tarefas em Máquinas Paralelas com Múltiplos Objetivos", In: XXXVI Simpósio Brasileiro de Pesquisa Operacional, v.1, p.1-11. (in Portuguese)

Celik, C. and Saricicek, I. (2009), "Tabu Search for Parallel Machine Scheduling with Job Splitting", In: Sixth International Conference on Information Technology, New Generations, p.183-188.

Dearing, P. M. and Henderson, R. A. (1984), "Assigning looms in a textile weaving operation with changeover limitations", Production and Inventory Management, Vol. 25, pp. 23-31.

Drummond, L. M. A.; Ochi, L. S. and Vianna, D. S. (2001), "An asynchronous parallel metaheuristic for the period vehicle routing problem", Future Generation Computer Systems Journal, Vol. 17, No.4, pp. 397-386.

Feo, T. A. and Resende, M.G.C. (1995), "Greedy randomized adaptive search procedure" Journal of Global Optimization, Vol. 6, pp. 109-133.

França, P. M.; Gendreau, M.; Laport, G. and Muller, F. (1996), "A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times", International Journal of Production Economics, Vol. 43, No. 2-3, pp. 79-89.

Mendes, A.; Müller, F. M.; França, P. M. and Moscato, P. (2002), "Comparing Meta-Heuristic Approaches For Parallel Machine Scheduling Problems", Production Planning & Control, Vol. 13, No. 2, pp. 143-154.

Mladenovic, N. and Hansen, P. (1997), "Variable Neighborhood Search", Computers and Operations Research, Vol. 24, pp. 1097-1100.

Ochi, L. S.; Drummond, L. M. A.; Victor, A. O. and Vianna, D. S. (1998), "A parallel evolutionary algorithm for solving the vehicle routing problem with heterogeneous fleet", Future Generation Computer Systems, Vol. 14, No. 5-6, pp. 285-292.

BJ Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

PETROBRAS. Petrobras' Homepage – link "Nossa História". Available at: <www.petrobras.com.br>. Access: June, 2008. (in Portuguese)

Resende, M. G. C. and Ribeiro, C. C. (2003), Greedy randomized adaptive search procedures. In: F. Glover and G. Kochenberger (eds.), (2003), Handbook of Metaheuristics. Boston: Kluwer Academic Publishers, 219-249.

Schrage, L. (1979), "A more portable FORTRAN random number generator", ACM Transactions on Mathematical Software, Vol. 5, pp. 132-138.

Sumichrast, R. and Baker, J. R. (1987), "Scheduling parallel processors: an integer linear programming based heuristic for minimizing setup time", International Journal of Production Research, Vol. 25, No. 5, pp. 761-771.

Vianna, D. S.; Ochi, L. S. and Drummond, L. M. A. (1999), "A parallel hybrid evolutionary metaheuristic for the period vehicle routing problem with heterogeneous fleet", Lecture Notes in Computer Science, 1388, pp. 216-225.

Zouba, M.; Baptiste, P. and Rebaine, D. (2009), "Scheduling identical parallel machines and operators within a period based changing mode", Computers and Operations Research, Vol. 36, No. 12, pp. 3231-3239.

## Biography

Dalessandro Soares Vianna is Professor in the Computer Science Department at Fluminense Federal University (UFF). He received his PhD degree in Combinatory Optimization from Catholic University of Rio de Janeiro (PUC-Rio), Brazil, in 2004. He received BSc and MSc degrees in Computer Science from Fluminense Federal University. His research interest includes multi-objective and mono-objective combinatorial optimization, metaheuristics, operational research and parallel processing.
Contact: dalessandro@pq.cnpq.br

Sandra Regina Coelho received her MSc degree in Operational Research and Computational Intelligence from Candido Mendes University (UCAM), Brazil, in 2007. She received her BSc degree in Computer Science from State University of São Paulo (UNESP). Her research interest includes operational research, metaheuristics and combinatorial optimization.
Contact: sandrarobl@yahoo.com.br

Marcilene de Fátima Dianin Vianna is Professor in the Economics Department at Fluminense Federal University (UFF). She is PhD student in the Natural Sciences Department of State University of the North Fluminense (UENF), Brazil. She received her MSc degree in Applied Mathematic from Catholic University of Rio de Janeiro (PUC-Rio), in 2001, and her BSc degree in Mathematic from State University of Maringá (UEM). Her research interest includes metabolomics, PCA, PLS, image processing and metaheuristics.
Contact: marcilenedianin@gmail.com

Brazilian Journal of Operations & Production Management
**Volume 7, Number 2, 2010, pp. 63-79**

## Article Info:

Received: May, 2010

Accepted: November, 2010