SPOLM 2007

ISSN 2175-6295 Rio de Janeiro- Brasil, 08 e 09 novembro de 2007.

# UMA HEURÍSTICA HÍBRIDA PARA O PROBLEMA DE ESCALONAMENTO DE EMBARCAÇÕES NO PORTO DE IMBETIBA - PETROBRAS

**Sandra Regina Coelho**
Universidade Candido Mendes – UCAM-Campos
Rua Anita Peçanha, 100, Pq. São Caetano, Campos dos Goytacazes, RJ 28040-320
sandrarobl@yahoo.com.br

**Dalessandro Soares Vianna**
Universidade Candido Mendes – UCAM-Campos
Rua Anita Peçanha, 100, Pq. São Caetano, Campos dos Goytacazes, RJ 28040-320
dalessandro@ucam-campos.br
Centro Federal de Educação Tecnológica – CEFET-Campos
Rua Dr. Siqueira, 273, Pq. Dom Bosco, Campos dos Goytacazes, RJ 28030-130
dalessandro@cefetcampos.br

**Fermín Alfredo Tang Montané**
Universidade Candido Mendes – UCAM-Campos
Rua Anita Peçanha, 100, Pq. São Caetano, Campos dos Goytacazes, RJ 28040-320
tang@ucam-campos.br

**Resumo**

A Petróleo Brasileiro S/A (PETROBRAS) é hoje a maior empresa brasileira no ramo de energia, atuando na exploração, produção, refino, comercialização e transporte de petróleo e seus derivados no Brasil e no exterior. A maior parte da produção de petróleo da empresa está concentrada na bacia de Campos, onde fica localizado o único porto próprio da empresa (porto de Imbetiba – Macaé/RJ). Através de estudos realizados neste porto, constatou-se a necessidade de otimizar, dentre outros, o problema de escalonamento de embarcações. Neste problema, a ordem de atendimento das embarcações deve ser definida e, de acordo com o material que cada uma carrega, as seguintes restrições devem ser respeitadas: cada embarcação tem um subconjunto de píeres que podem atendê-las; cada uma tem uma prioridade de atendimento; e cada uma tem um horário mínimo para atendimento, ou seja, antes deste horário a embarcação não pode ser atendida. Este problema pode ser associado ao "Problema de Escalonamento de Tarefas em Máquinas Paralelas", onde as máquinas são os píeres e as tarefas são as embarcações.

Este trabalho propõe uma heurística GRASP híbrida que utiliza a técnica VND na etapa de busca local. Três estruturas de vizinhança foram implementadas: *Exchange*, *Interchange* e *Relocation*. Os resultados computacionais mostram que a heurística proposta é mais eficiente que heurísticas GRASP tradicionais quando o custo das soluções obtidas é

comparado e, também, quando os tempos computacionais exigidos são comparados.
**Palavras-Chaves:** Problema de Escalonamento de Tarefas em Máquinas Paralelas; GRASP; VND; Otimização Combinatória.

**Abstract**

Petróleo Brasileiro S/A (PETROBRAS) is nowadays the biggest Brazilian company in energy business. It acts at the exploration, production, refine, 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. Studies made at this port show the need of optimize, among others, the problem of towboat scheduling. In this problem, the order of towboat granting 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 granted; each one has a granting priority; and each one has a minimal time where it can be granted, that is, before this time it can not be granted. This problem can be associated to the "Job Scheduling Problem on Parallel Machines", where the machines are the piers and the jobs are the towboats.

This paper proposes a hybrid GRASP algorithm, which uses the VND technique as local search. Three neighborhood structures were implemented: Exchange, Interchange and Relocation. The computational results show that the proposed algorithm is more efficient than the traditional GRASP algorithms when the obtained solution cost is evaluated and, also, when the computational time required is compared.
**Keywords:** Job Scheduling Problem on Parallel Machines; GRASP; VND; Combinatorial Optimization.

## 1. INTRODUÇÃO

A Petróleo Brasileiro S/A (PETROBRAS) iniciou-se no século XX e é hoje a maior empresa brasileira do ramo de energia, atuando na exploração, produção, refino, comercialização e transporte de petróleo e seus derivados no Brasil e no exterior. A estrutura organizacional regula o funcionamento instituindo quatro áreas de negócio: Exploração & Produção, Abastecimento, Gás & Energia e Internacional, e duas de apoio: Financeira e Serviços, além das unidades corporativas ligadas diretamente ao presidente (PETROBRAS, 2006).

Atualmente, a empresa possui 95 plataformas de produção – sendo 72 fixas e 23 flutuantes –, 16 refinarias, 30.318 quilômetros de dutos e 6.154 postos de combustíveis espalhados pelo território nacional, dos quais 631 são próprios (PETROBRAS, 2006).

A maior parte da produção de petróleo da empresa está concentrada na bacia de Campos, onde fica localizado o único porto próprio da empresa (porto de Imbetiba – Macaé/RJ). Através desse porto é feito todo o abastecimento das plataformas da bacia.

Através de estudos realizados no porto de Imbetiba, constatou-se que a empresa apresenta algumas tarefas a serem otimizadas. Se for realizado um estudo em cima destas tarefas, algumas soluções podem ser apresentadas de forma a se ter um melhor aproveitamento das embarcações e do porto de Macaé. O ganho poderia ser substancial para a PETROBRAS e para as empresas contratadas que utilizam o porto de Imbetiba.

Dentre essas tarefas, destaca-se o problema de otimizar o tempo de espera das embarcações no porto. Neste problema, a ordem de atendimento deve ser decidida e, de

acordo com o material que cada embarcação carrega, as seguintes restrições devem ser respeitadas:

- cada embarcação possui um subconjunto de píeres que pode atendê-la. Isso ocorre devido a cada píer estar preparado para o embarque/desembarque de um conjunto de materiais;
- cada embarcação tem uma prioridade de atendimento. A prioridade é dada de acordo com a urgência do material, do tipo de material: perecível ou não, priorizando o atendimento a uma plataforma que no momento está com um nível de produção maior, dentre outros fatores;
- tempo mínimo para atendimento (*release time*), que representa o momento que a plataforma já se encontra a disposição.

Este problema pode ser associado ao Problema de Escalonamento de Tarefas em Máquinas Paralelas – PETMP – (ARROYO & RIBEIRO, 2004; MENDES *et al.*, 2002) , onde um conjunto de tarefas, cada uma com um tempo de execução, deve ser organizado para execução em um conjunto de máquinas idênticas, de tal forma que o tempo total de execução seja minimizado. No problema estudado, as tarefas são as embarcações e as máquinas são os píeres, que não são idênticos – cada um preparado para atender a demanda de um subconjunto de materiais. Além disso, as restrições de prioridade e de *release time* devem ser consideradas. A existência deste conjunto de restrições deve-se ao fato do porto de Imbetiba ser um porto privado e pequeno, possuindo regras próprias diferentes das estabelecidas em grandes portos. Este fato tornou impraticável comparar os estudos realizados com outros trabalhos já realizados para outros portos.

Este trabalho propõe um algoritmo GRASP híbrido que utiliza a técnica VND na fase de busca local. Três estruturas de vizinhança forma implementadas: *Exchange*, *Interchange* e *Relocation*. A literatura sobre a classe específica de PETMP tratada neste trabalho é praticamente nula. O algoritmo proposto é comparado com algoritmos GRASP tradicionais propostos e publicados (VIANNA & COELHO, 2006) pelos próprios autores deste trabalho. Os resultados computacionais mostram que o algoritmo proposto é mais eficiente do que os outros algoritmos quando a qualidade da solução obtida é comparada e, também, quando os tempos computacionais são comparados.

O restante deste trabalho está organizado da seguinte maneira: na Seção 2 são apresentados o problema PETMP, a literatura relacionada e a classe do PETMP abordada neste trabalho. O algoritmo GRASP híbrido utilizando a técnica VND na etapa de busca local é detalhado na Seção 3. Na Seção 4 são apresentados os resultados computacionais. Por fim, são apresentadas a conclusão e as referências bibliográficas.

## 2. PROBLEMA DE ESCALONAMENTO DE TAREFAS EM MÁQUINA PARALELAS (PETMP)

De acordo com Mendes *et al.* (2002), o PETMP pode ser definido da seguinte maneira: um conjunto de *n* tarefas deve ser organizado em *m* máquinas paralelas idênticas com o objetivo de minimizar o *makespan* – tempo necessário para as *n* tarefas completarem a sua execução. Cada máquina possui um tempo de *setup* dependente, ou seja, o tempo necessário para preparar a máquina para atender uma tarefa depende de qual tarefa acabou de ser executada por esta máquina.

Na literatura relacionada ao PETMP encontra-se: um modelo de programação linear inteira, proposto por Dearing e Henderson (1984), para a alocação do tear em uma tecelagem; um método heurístico, proposto por Sumicharst e Baker (1987), baseado na solução de uma

série de subproblemas de programação inteira 0-1 que melhora 2 resultados de Dearing e Henderson (1984); uma heurística busca tabu é proposta por França *et al.* (1996); Arroyo e Ribeiro (2004) propõem um algoritmo genético para o PETMP com múltiplos objetivos; e uma heurística busca tabu, proposta por Mendes *et al.* (2002), para o PETMP não preemptivo.

A classe de problema abordada neste trabalho, PETMP*, pode ser definida da seguinte maneira: um conjunto de *n* tarefas deve ser organizado em *m* máquinas **não** idênticas. O tempo de *setup* é independente, ou seja, é o mesmo independente de quem foi a tarefa que acabou de ser executada. Sendo assim, os tempos de *setup* são desconsiderados neste trabalho. Cada tarefa possui um subconjunto de máquinas que podem executá-la, um tempo mínimo para seu atendimento (*release time*) e uma prioridade.

Cada tarefa *i* possui uma prioridade $p_i$. Quanto maior for o valor de $p_i$, mais prioritária é a tarefa *i*. Cada tarefa *i* também possui um *release time* $rt_i$. Sendo $at_i$ o momento no qual a tarefa *i* iniciou efetivamente seu atendimento, pode-se calcular o tempo de espera, $te_i$, da tarefa *i* da seguinte forma: $te_i = at_i - rt_i$. O objetivo ao se resolver o PETMP* é minimizar o tempo de espera total, levando em conta a prioridade de cada tarefa. O tempo de espera total pode ser calculado da seguinte forma:

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

Na literatura, Vianna e Coelho (2006) propuseram algumas heurísticas GRASP para o problema PETMP*, onde três algoritmos de construção e duas vizinhanças foram desenvolvidos. Neste trabalho é proposto um algoritmo GRASP híbrido utilizando a técnica VND na fase de busca local. Este algoritmo utiliza, na fase de construção, o algoritmo **ETQM1** – será descrito em detalhes na Subseção 3.1 – proposto em (VIANNA & COELHO, 2006). Este algoritmo construtivo se destacou nos experimentos realizados por Vianna e Coelho (2006), onde este foi comparado com outros dois algoritmos, **ETP** e **ETQM2**, também propostos no mesmo trabalho.

Na etapa de busca local são usadas três estruturas de vizinhanças: duas utilizadas também por Vianna e Coelho (2006) – ***Exchange*** e ***Interchange***; e mais uma implementada neste trabalho, ***Relocation***, que será descrita em detalhes na Subseção 3.2.3. O algoritmo VND implementado será comentado na Subseção 3.2.

## 3. HEURÍSTICA GRASP+VND PROPOSTA

GRASP – *Greedy Randomized Adaptive Search Procedure* – (FEO & RESENDE, 1995; RESENDE & RIBEIRO, 2003) é uma metaheurística de múltiplas partidas, na qual cada iteração consiste de duas fases: construção e busca local. A primeira fase constrói uma solução viável utilizando um algoritmo guloso randomizado, cuja vizinhança é investigada até que um ótimo local seja encontrado durante a fase de busca local. A melhor solução entre todas é definida como resultado.

Neste trabalho é proposto um algoritmo GRASP híbrido que utiliza a técnica VND – *Variable Neighborhood Descent* – (MLADENOVIC & HANSEN, 1997) na etapa de busca local. Este algoritmo utiliza na etapa de construção o algoritmo **ETQM1** (VIANNA & COELHO, 1996), que será descrito em detalhes na Subseção 3.1, e na etapa de busca local a técnica VND utiliza três estruturas de vizinhança que serão comentadas na Subseção 3.2.

### 3.1. HEURÍSTICA CONSTRUTIVA ETQM1

Nesta heurística construtiva é considerada, primeiramente, a quantidade de máquinas que podem executar determinada tarefa, tomando-se como fator decisório as tarefas que têm o menor número de máquinas para executá-las. O intuito desta estratégia é diminuir o tempo de espera total, uma vez que as tarefas com um menor número de possíveis máquinas que podem atendê-las são atendidas na frente.

Dessa forma, inicialmente, as tarefas são classificadas, em ordem crescente pela quantidade de máquinas que podem atendê-las. Assim, as tarefas que só podem ser atendidas por uma máquina estarão no início da fila. Quando mais de uma tarefa possui a mesma quantidade de máquinas para atendê-la, o desempate é dado pela prioridade das tarefas (considerando-se a ordem decrescente das prioridades).

Para que a cada iteração do algoritmo GRASP uma solução inicial diferente seja construída, em vez de sempre escolher a primeira tarefa da fila, escolhe-se, aleatoriamente, uma entre as $\alpha$ primeiras tarefas, onde $\alpha$ é um parâmetro de entrada do algoritmo.

Ao se inserir uma tarefa *i* na máquina, deve-se escolher a melhor posição considerando todas as posições disponíveis na máquina, a partir do *release time* da tarefa *i*, de forma que o tempo de espera total seja o menor possível. Quando uma tarefa puder ser executada por mais de uma máquina, escolhe-se a máquina onde a tarefa não ocasionar desperdício de tempo ou ocasionar o menor tempo de espera possível.

```
Procedimento ETQM1 ( α )
Entrada
    α – quantidade de tarefas a ser feita a aleatoriedade.
Saída
    s – solução construída.
Início
01. Ordene crescentemente a lista L de tarefas pela quantidade de máquinas que pode atender e, em
    caso de empate, pela prioridade;
02. Para i ← 1 até n faça
03.     b ← escolha, aleatória, de uma das α primeiras tarefas de L ainda não atendidas;
04.     best_pos ← ∞;
05.     Para cada máquina j que pode atender a tarefa b faça
06.         y ← melhor posição na máquina j para inserir a tarefa b, ou seja, a posição que causa o
            menor tempo de espera;
07.         Se y < best_pos então
08.             best_pos ← y;
09.             best_mac ← j;
10.         Fim-se
11.     Fim-para
12.     Insira a tarefa b na posição best_pos da máquina best_mac da solução s;
13.     Calcule o novo tempo de espera acumulado da máquina best_mac;
14. Fim-para
15. Retorne s;
Fim-ETQM1
```

Figura 1. Algoritmo **ETQM1** desenvolvido em (VIANNA & COELHO, 2006).

Para se inserir uma tarefa na máquina, consideram-se todas as posições livres da máquina de forma que será considerada a posição que causar o menor tempo de espera final na máquina.

A Figura 1 apresenta o algoritmo **ETQM1**, que recebe como parâmetro de entrada o grau de aleatoriedade, $\alpha$, e retorna como saída a solução *s* construída. Na linha 1, a lista com todos as tarefas é ordenada crescentemente pela quantidade de máquinas que pode atender cada tarefa e pela prioridade. O laço nas linhas 2-14 garante que todas as tarefas serão inseridas na solução *s*. Na linha 4, uma tarefa *b* é escolhida aleatoriamente entre as $\alpha$ primeiras da lista ainda não inseridas na solução *s*. O laço nas linhas 5-11 escolhe, nas máquinas que podem atender a tarefa *b*, a melhor posição para inserir a tarefa *b*. Esta inserção é realizada na linha 12. Na linha 14, a solução *s* é retornada.

### 3.2. BUSCA LOCAL USANDO VND

O algoritmo da Figura 2 apresenta o pseudocódigo do algoritmo VND desenvolvido que utiliza três estruturas de vizinhança: *Exchange*, *Interchange* e *Relocation*, que serão descritas em detalhes nas Subseções 3.2.1, 3.2.2 e 3.2.3, respectivamente. Este algoritmo recebe como parâmetro de entrada, além das três vizinhanças, uma solução viável $s_0$ a ser refinada. A primeira vizinhança a ser analisada é a vizinhança $N_1$ (linha 2). O laço nas linhas 3-11 garante que todas as estruturas de vizinhanças serão analisadas. Se após a aplicação da busca local na vizinhança $N_k$, a solução encontrada for melhor que a original, a busca reinicia (linha 7) com a vizinhança $N_1$. Caso contrário, a busca continua na próxima vizinhança (linha 9). Na linha 12 a solução *s* refinada é retornada.

```
Procedimento VND ( s0, N1, N2, N3 )
Entrada
    x0 - uma solução viável para o problema;
    N1 – estrutura de vizinhança Exchange;
    N2 – estrutura de vizinhança Interxchange;
    N3 – estrutura de vizinhança Relocation.
Saída
    s – melhor solução encontrada.
Início
01. s ← s0;
02. k ← 1;
03. Enquanto  k ≤ 3 faça
04.     Aplique uma busca local em s usando a vizinhança Nk. Seja s' o ótimo local;
05.     Se f(x') < f(x) então
06.         s ← s';
07.         k ← 1;
08.     Senão
09.         k ← k+1;
10.     Fim-se
11. Fim-enquanto
12.  Retorne s;
Fim-VND
```

Figura 2. Procedimento **VND** utilizado.

#### 3.2.1. Vizinhança *Exchange*

Neste procedimento de busca local, a partir de uma solução inicial, considera-se a troca de

posições entre duas embarcações que se encontram na fila de atendimento de um mesmo píer. A Figura 3 apresenta o pseudocódigo do procedimento de busca local analisando esta vizinhança, que recebe como parâmetro de entrada a solução *s* a ser refinada. O laço nas linhas 2-20 garante que a busca continuará enquanto existir um vizinho melhor que a solução corrente *s*. O laço nas linhas 4-13 procura a melhor troca entre tarefas de uma mesma máquina que pode ser realizada. Se a melhor troca levar a uma solução melhor que a solução corrente, esta troca é realizada na linha 15. Caso contrário o procedimento é encerrado. A solução *s* refinada é retornada na linha 21.

```
Procedimento LS_Exchange ( s )
Entrada
   s – solução viável.
Saída
   s – solução refinada
Início
01. Terminou ← false;
02. Enquanto não terminou faça
03.     Melhor_troca  ← ∞;
04.     Para cada tarefa i faça
05.         Para cada tarefa j (j ≠ i) da mesma máquina de i faça
06.             c ← custo adicional da troca de posição das tarefas i e j;
07.             Se c < Melhor_troca então
08.                 Melhor_troca ← c;
09.                 Melhor_i ← i;
10.                 Melhor_j ← j;
11.             Fim-se
12.         Fim-para
13.     Fim-para
14.     Se Melhor_troca < 0 então
15.         Realize a troca das tarefas Melhor_i e Melhor_j;
16.         Avalie a solução s após a troca;
17.     Senão
18.         Terminou ← true;
19.     Fim-se
20. Fim-Enquanto
21. Retorne s;
Fim- LS_Exchange
```

Figura 3. Algoritmo **LS_Exchange**.

### 3.2.2. Vizinhança *Interchange*

Neste procedimento de busca local, a partir de uma solução inicial, considera-se a troca de posições entre duas embarcações que se encontram na fila de atendimento de píeres diferentes. A Figura 4 apresenta o pseudocódigo do procedimento de busca local utilizando esta vizinhança, que recebe como parâmetro de entrada a solução *s* a ser refinada. O laço nas linhas 2-20 garante que a busca continuará enquanto existir um vizinho melhor que a solução corrente *s*. O laço nas linhas 4-13 procura a melhor troca entre tarefas de máquinas diferentes que pode ser realizada. Se a melhor troca levar a uma solução melhor que a solução corrente,

esta troca é realizada na linha 15. Caso contrário o procedimento é encerrado. A solução *s* refinada é retornada na linha 21.

```
Procedimento LS_Interchange ( s )
Entrada
   s – solução viável.
Saída
   s – solução refinada
Início
01. Terminou ← false;
02. Enquanto não terminou faça
03.     Melhor_troca  ← ∞;
04.     Para cada tarefa i faça
05.         Para cada tarefa j de uma outra máquina que também pode atender a tarefa i faça
06.             c ← custo adicional da troca de posição das tarefas i e j;
07.             Se c < Melhor_troca então
08.                 Melhor_troca ← c;
09.                 Melhor_i ← i;
10.                 Melhor_j ← j;
11.             Fim-se
12.         Fim-para
13.     Fim-para
14.     Se Melhor_troca < 0 então
15.         Realize a troca das tarefas Melhor_i e Melhor_j;
16.         Avalie a solução s após a troca;
17.     Senão
18.         Terminou ← true;
19.     Fim-se
20. Fim-Enquanto
21. Retorne s;
Fim- LS_ Interchange
```

Figura 4. Algoritmo **LS_Interchange**.

#### 3.2.3. Vizinhança *Relocation*

Neste procedimento de busca local, a partir de uma solução inicial, considera-se a migração de uma tarefa que está em um berço *j* para outro berço *k* que também é capaz de atendê-la. A Figura 5 apresenta o pseudocódigo do procedimento de busca local utilizando esta vizinhança, que recebe como parâmetro de entrada a solução *s* a ser refinada. O laço nas linhas 2-21 garante que a busca continuará enquanto existir um vizinho melhor que a solução corrente *s*. O laço nas linhas 4-14 procura a melhor realocação entre tarefas de máquinas diferentes. Se a melhor troca levar a uma solução melhor que a solução corrente, esta troca é realizada na linha 16. Caso contrário o procedimento é encerrado. A solução *s* refinada é retornada na linha 22.

```
Procedimento LS_Relocation ( s )
Entrada
   s – solução viável.
Saída
```

```
    s – solução refinada
Início
01. Terminou ← false;
02. Enquanto não terminou faça
03.     Melhor_realocação ← ∞;
04.     Para cada tarefa i faça
05.         Para cada posição pos de uma outra máquina mac_j que também pode atendê-la faça
06.             c ← custo adicional da retirada da tarefa i da máquina atual e sua inserção na
                posição pos da máquina mac_j;
07.             Se c < Melhor_realocação então
08.                 Melhor_ realocação ← c;
09.                 Melhor_i ← i;
10.                 Melhor_mac_j ← mac_j;
11.                 Melhor_pos ← pos;
12.             Fim-se
13.         Fim-para
14.     Fim-para
15.     Se Melhor_troca < 0 então
16.         Realize a retirada da tarefa Melhor_i de sua máquina atual e a sua inserção na posição
            Melhor_pos da máquina Melhor_mac_j;
17.         Avalie a solução s após a realocação;
18.     Senão
19.         Terminou ← true;
20.     Fim-se
21. Fim-Enquanto
22. Retorne s;
Fim- LS_ Relocation
```

Figura 5. Algoritmo **LS_ Relocation**.

## 4. RESULTADOS COMPUTACIONAIS

Todos os experimentos computacionais deste trabalho foram realizados em um Notebook Compaq Presario R3000 com processador Atlhon 3200 e 520Mb de memória RAM.

O algoritmo proposto, bem como os algoritmos GRASP usados para efeito de comparação, foram implementados utilizando a linguagem de programação C.

Na Subseção 4.1 serão apresentados os algoritmos GRASP que foram implementados para validação do algoritmo proposto. Os problemas testes utilizados nos experimentos serão discutidos na Subseção 4.2. Na Subseção 4.3 serão mostrados os experimentos realizados.

### 4.1. HEURÍSTICAS GRASP

Foram implementados, para efeito de comparação, três algoritmos GRASP. Todos eles utilizam o procedimento **ETQM1** na etapa de construção e respeitam o procedimento GRASP padrão apresentada na Figura 6. O que diferencia estas estratégias é a estrutura de vizinhança utilizada na etapa de busca local.

```
Procedimento GRASP ( N_iter, α )
Entrada
    N_iter - número de iterações GRASP;
```

$\alpha$ - define o grau de aleatoriedade do algoritmo.
**Saída**
*s* – melhor solução encontrada.
**Início**
01. **Para** *i* **de** 1 **até** *N_iter* **faça**
02. **Início**
03. *s*' ← **Construção** ($\alpha$);
04. s'' ← **BuscaLocal** (*s*');
05. **Se** *s*'' for a melhor solução até o momento **então**
06. *s* ← *s*'';
07. **Fim-para**
08. **Retorne** *s*;
**Fim-GRASP**

Figura 6. Algoritmo GRASP padrão.

Os três algoritmos GRASP implementados são:

- **GRASP_Ex** – utiliza o procedimento **LS_Exchange** na etapa de busca local;
- **GRASP2_Inter** – utiliza o procedimento **LS_Interchange** na etapa de busca local;
- **GRASP2_Re** – utiliza o procedimento **LS_Relocation** na etapa de busca local.

### 4.2. PROBLEMAS TESTES

Para realizar uma comparação entre o algoritmo proposto e os algoritmos GRASP, descritos na Subseção 4.1, foram usados os 16 problemas testes criados por Vianna e Coelho (2006), que tratava do mesmo problema abordado neste trabalho, mas usando parâmetros que se aproximassem de um caso real do problema de otimizar o tempo de espera das embarcações no porto de Imbetiba (Macaé/RJ), problema que motivou este trabalho. Estes problemas testes possuem $m$ = 5 a 8 píeres (máquinas) e o intervalo de tempo a ser escalonado variando de $T$ = 48 a 120 horas. A Tabela 1 mostra, para cada problema teste, o número de píeres do problema ($m$) e o tempo total para escalonamento ($T$).

Em (VIANNA & COELHO, 2006), os outros parâmetros destes problemas testes foram definidos da seguinte forma. O *release time* (*rt*) de cada embarcação foi definido aleatoriamente entre 0 e ($T$-1) horas. O tempo de embarque/desembarque de uma embarcação (*at*) foi definido aleatoriamente entre 1 e 24 horas. A quantidade de embarcações, *n*, do problema foi definida pela seguinte fórmula: ($m$*$T$)/12, onde o denominador representa o tempo médio de atendimento (embarque/desembarque) de uma embarcação (tarefa) . A prioridade de atendimento de cada embarcação foi definida aleatoriamente entre os valores de 0 a 5.

Tabela 1 – Problemas testes.

| **Problema teste** | **Número de píeres ($m$)** | **Intervalo de tempo ($T$) em horas** |
|---|---|---|
| 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 |

### 4.3. EXPERIMENTOS REALIZADOS

No primeiro experimento realizado, o algoritmo **GRASP+VND** proposto e cada algoritmo GRASP descrito na Subseção 4.1 foram executados durante *N_iter*=1000 iterações para cada problema teste descrito na Tabela 1. Este processo foi repetido cinco vezes, variando a semente de geração de números aleatórios. Para a geração de números aleatórios neste trabalho foi usada uma versão na linguagem de programação C do gerador descrito em [Schrage, 1979].

A Tabela 2 apresenta os custos (tempo de espera total) e os tempos (em segundos) obtidos, para cada problema teste, pelo algoritmo proposto e pelos outros algoritmos GRASP. Nesta tabela foram destacados, em negrito, os melhores custos obtidos para cada problema teste. O algoritmo **GRASP+VND** obteve os melhores custos para 12 dentre os 16 problemas testes. Para os outros 4, o algoritmo **GRASP_Re**, que utiliza a vizinhança *Relocation*, obteve os melhores custo. Quando os tempos computacionais são comparados, o algoritmo **GRASP+VND** sempre obteve tempos melhores. Isto acontece pois, utilizando as diferentes estruturas de vizinhança, a convergência na etapa de busca local é mais rápida.

Tabela 2. Resultados do primeiro experimento.

| *Problema* | **GRASP_Ex** | | **GRASP_Inter** | | **GRASP_Re** | | **GRASP+VND** | |
|---|---|---|---|---|---|---|---|---|
| *Teste* | *Custo* | *Tempo* | *Custo* | *Tempo* | *Custo* | *Tempo* | *Custo* | *Tempo* |
| 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 |

Uma vez que o algoritmo **GRASP+VND** sempre apresentou um tempo menor no primeiro experimento, foi realizado um segundo experimento, onde foi verificado como os outros algoritmos se comportariam caso interrompessem a sua execução quando atingisse o tempo computacional gasto pelo algoritmo **GRASP+VND**. Deste modo, a condição de parada dos algoritmos GRASP foi alterada para o tempo gasto pelo algoritmo **GRASP+VND**. Tabela 3 apresenta os custos obtidos por cada algoritmo.

Tabela 3. Resultados do segundo experimento.

| *Problema* | **GRASP_Ex** | **GRASP_Inter** | **GRASP_Re** | **GRASP+VND** |
|---|---|---|---|---|
| *Teste* | *Custo* | *Custo* | *Custo* | *Custo* |
| 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** |

Neste novo experimento percebe-se ainda mais a superioridade do algoritmo **GRASP+VND**. Em 15 dos 16 problemas testes, o custo obtido pelo algoritmo proposto foi melhor do queo dos outros algoritmos. Apenas na instância “Ins7-96” o algoritmo proposto foi superado por 0,3%.

## 5. CONCLUSÃO

Este trabalho surgiu da necessidade de se resolver o problema de minimizar o tempo de espera das embarcações no porto de Imbetiba (Macaé/RJ). Este porto, por ser um porto pequeno e exclusivo da PETROBRAS, possui regras próprias que impedem a comparação com outros estudos realizados para outros portos. Este problema foi associado ao problema de escalonamento de tarefas em máquinas paralelas (ARROYO & RIBEIRO, 2004; DEARING & HENDERSON, 1984; FRANÇA *et al.*, 1996; MENDES *et al.*, 2002; SUMICHRAST & BAKER, 1987), que é NP-difícil. Desta maneira, uma boa estratégia para resolvê-lo é fazer o uso de metaheurísticas (DRUMMOND *et al.*, 2001; OCHI *et al.*, 1998; VIANNA *et al.*, 1999).

O algoritmo GRASP híbrido proposto, **GRASP+VND**, mostrou-se, nos experimentos realizados, eficiente para a classe do Problema de Escalonamento de Tarefas em Máquinas Paralelas abordada. Em um dos experimentos realizados, onde o mesmo tempo de execução foi dado a todos os algoritmos, o algoritmo proposto obteve o melhor resultado para 15 dos 16 problemas testes, sendo superado apenas em uma instância por uma diferença pequena. Isso

demonstra que utilizar um procedimento que utiliza diferentes estruturas de vizinhança, como é o caso do VND, pode trazer benefícios quando incorporada a outra metaheurística.

## AGRADECIMENTOS

Este trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq.

## REFERÊNCIAS BIBLIOGRÁFICAS

ARROYO, J. E. C., RIBEIRO, R. L. P. 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, 2004.

DEARING, P. M., HENDERSON, R. A. Assigning looms in a textile weaving operation with changeover limitations. *Production and Inventory Management* 25, 23-31, 1984.

DRUMMOND, L. M. A., OCHI, L. S., VIANNA, D. S. An asynchronous parallel metaheuristic for the period vehicle routing problem. *Future Generation Computer Systems Journal* 17(4), 397-386, 2001.

FEO, T.A., RESENDE, M.G.C. Greedy randomized adaptive search procedure. *Journal of Global Optimization* 6, 109-133, 1995.

FRANÇA, P. M., GENDREAU, M., LAPORT, G., MULLER, F. A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *International Journal of Production Economics* 43, 79-89, 1996.

MENDES, A., MÜLLER, F. M., FRANÇA, P. M., MOSCATO, P. Comparing Meta-Heuristic Approaches For Parallel Machine Scheduling Problems. *Production Planning & Control* 13(2), 143-154, 2002.

MLADENOVIC, N.; HANSEN, P. Variable Neighborhood Search. *Computers and Operations Research* 24, 1097-1100, 1997.

OCHI, L. S., DRUMMOND, L. M. A., VICTOR, A. O., VIANNA, D. S. A parallel evolutionary algorithm for solving the vehicle routing problem with heterogeneous fleet. *Future Generation Computer Systems* 14, 285-292, 1998.

PETROBRAS. Disponível em: <www.petrobras.com.br>. Acessado em maio de 2006.

RESENDE, M. G. C., RIBEIRO, C. C. Greedy randomized adaptive search procedures. In F. Glover e G. Kochenberger (eds.), *Handbook of Metaheuristics*. Kluwer, 219-249, 2003.

SCHRAGE, L. A more portable FORTRAN random number generator, *ACM Transactions on Mathematical Software* 5, 132-138, 1979.

SUMICHRAST, R., BAKER, J. R. Scheduling parallel processors: an integer linear programming based heuristic for minimizing setup time, *International Journal of Production Research* 25(5), 761-771, 1987.

VIANNA, D. S. ; COELHO, S. R. Otimização do Tempo de Espera das Embarcações no Porto de Imbetiba - Petrobrás. In: *XXXVIII Simpósio Brasileiro de Pesquisa Operacional*, Goiânia, 1814-1824, 2006.

VIANNA, D. S., OCHI, L. S., DRUMMOND, L. M. A. A parallel hybrid evolutionary metaheuristic for the period vehicle routing problem with heterogeneous fleet. *Lecture Notes in Computer Science* 1388, 216-225, 1999.