OTIMIZAÇÃO DO TEMPO DE ESPERA DAS EMBARCAÇÕES NO PORTO DE IMBETIBA – PETROBRÁS

Dalessandro Soares Vianna
Universidade Candido Mendes – UCAM-Campos,
Rua Anita Pessanha, 100, Parque São Caetano, Campos dos Goytacazes, RJ 28040-320.
dalessandro@ucam-campos.br

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

RESUMO

A Petrobras é hoje a maior empresa brasileira do ramo de energia e atua na exploração, produção, refino, comercialização e transporte de petróleo e seus derivados no Brasil e no exterior. A maior parte desta produção está concentrada na bacia de Campos, onde fica localizado o único porto próprio da empresa – porto de Imbetiba. Através desse porto é feito todo o abastecimento das plataformas da bacia. O objetivo principal deste trabalho é desenvolver um algoritmo eficiente para o minimizar o tempo de espera das embarcações que realizam o abastecimento das plataformas. Por se tratar de um porto próprio e pequeno, possui regras próprias, portanto, não foi possível basear o estudo em trabalhos já realizados para outros portos. Para efeito de comparação, são propostos neste trabalho diferentes algoritmos GRASP. O que apresentar, dentre eles, os melhores resultados será definido como o que melhor se adapta ao problema em questão.

PALAVRAS CHAVE. Escalonamento de tarefas. Metaheurística GRASP. Otimização combinatória. Metaheurísticas.

ABSTRACT

Nowadays, the Petrobras is the biggest Brazilian company in energy business. It acts at the exploration, production, refine, commercialization and transport of petroleum by products in Brazil and another countries. Most of the petroleum production is concentrated in the Campos basin, where the company port – port of Imbetiba – is located. All the platforms supply is done using this port. The main goal of this work is to develop an efficient algorithm to minimize the wait time of the ships that carry out the platforms supply. The port of Imbetiba is an own and small port, so it has own rules that not permit to compare this work with another that was done for another port. To make a comparison, in this work are proposed different GRASP algorithms. The solutions obtained by the better GRASP algorithm will be kept as the result.

KEYWORDS. Task scheduling. GRASP metaheuristic. Combinatory optimization. Metaheuristic.

1- INTRODUÇÃO

A Petróleo Brasileiro S/A (Petrobras) iniciou-se no século XX e é hoje a maior empresa brasileira do ramo de energia e atua 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.

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 destas tarefas, algumas soluções poderiam ser apresentadas de forma a se ter um melhor aproveitamento das embarcações e do porto de Macaé. O ganho poderia estar representando valores altíssimos para a Petrobrás 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. O objetivo principal deste trabalho é desenvolver um algoritmo eficiente para este problema. Para isso, serão propostos algoritmos que realizem o escalonamento das embarcações a serem atendidas de tal forma que o tempo de espera seja minimizado, respeitando as prioridades associadas a cada embarcação.

Por se tratar de um porto próprio e pequeno, possui regras próprias, portanto, não foi possível basear o estudo em trabalhos já realizados para outros portos, visto que as regras estabelecidas pela Petrobrás para o porto de Imbetiba são muito diferentes das regras estabelecidas para grandes portos. Para efeito de comparação, são propostos neste trabalho diferentes algoritmos GRASP [Arroyo *et al.*, 2005; Feo & Resende, 1995; Ribeiro & Vianna, 2005]. O que apresentar, dentre eles, os melhores resultados será definido como o que melhor se adapta ao problema em questão.

A otimização realizada neste trabalho pode representar a economia de um montante altíssimo para a Petrobrás e as empresas que usam o porto de Imbetiba para transporte de materiais, visto que o custo diário com o aluguel das embarcações é muito alto.

Na próxima seção serão apresentadas as características do problema da fila de espera das embarcações no porto de Imbetiba. Na Seção 3 serão discutidos os algoritmos GRASP propostos. Os resultados obtidos com os algoritmos propostos serão apresentados na Seção 4. Por fim, são apresentadas as conclusões e as referências bibliográficas.

2 – PROBLEMA ABORDADO

Esta seção tem como objetivo mostrar como são realizadas as operações portuárias no porto de Imbetiba.

Parte dos estudos para esse trabalho está sendo baseada nos relatórios apresentados para a construção do modelo de simulação de operações portuárias já existente atualmente na Petrobras

[CE_UFRJ, 1997a; CE_UFRJ, 1997b], adaptando-se às mudanças realizadas na empresa após o levantamento feito anteriormente.

As regras aplicadas ao porto de Imbetiba, por se tratar de um porto pequeno e que atende somente as demandas da Petrobras, obedecem a critérios próprios, estabelecidos pela própria empresa, o que não permite basear o estudo realizado neste trabalho com algoritmos desenvolvidos para outros portos, vistos que estes obedecem a regras padrões dos grandes portos, como, por exemplo, considerar a ordem de chegada das embarcações ou multa por atraso no atendimento.

Nas próximas subseções serão discutidos alguns detalhes importantes sobre o funcionamento do Porto de Imbetiba.

2.1. Embarcações

A análise das embarcações baseou-se no sistema de transporte marítimo e na situação portuária atualmente existente. A partir destas análises, as embarcações foram agregadas em três tipos distintos:

- **Suprimento** – são as embarcações que utilizam guindastes, empilhadeiras e carretas para carga de materiais, tanto de convés, quanto tubo. Esses tipos de embarcações fazem, também, abastecimento de água, óleo e, algumas, de lama.
- **Graneleiras** – são aquelas que carregam somente materiais do tipo granel, tais como cimento, baritina e bentonita. O carregamento é feito utilizando-se mongotes e, eventualmente, algumas, utilizam guindastes para carregamento de materiais.
- **Exclusivas** (ou especiais) – é o tipo de embarcação que fica mais tempo no píer (local de atracamento das embarcações), usam guindaste em menor intensidade ou, quando o usam, é feito com outros propósitos que não o de carga de materiais (ex: embarcações de pesquisa) e têm operações diferenciadas das demais embarcações.

As embarcações de suprimento podem ocupar qualquer berço, visto que todos os berços são equipados com guindastes e mangotes de fluídos. Portanto, esse tipo de embarcação, pode atracar no primeiro berço que se tornar disponível no momento do embarque/desembarque de material da embarcação.

As embarcações de granéis têm prioridade nos berços que possuem pontos de abastecimento de granéis.

As embarcações exclusivas podem atracar em qualquer berço quando se trata de abastecimento de água e óleo diesel, ou material leve e somente em certos berços quando se trata de embarque/desembarque de materiais de ancoragem (âncora, estaca, torpedo, etc), que necessitam de guindastes especiais.

Sendo assim, serão definidas, a princípio, prioridades de atendimento para cada embarcação, conforme restrições detalhadas mais adiante, e a escolha para que uma embarcação atraque num determinado berço dependerá dessa prioridade e do tipo de berço disponível (visto que determinadas embarcações só podem atracar em alguns berços específicos, como já citado).

2.2. Atendimentos

Todos os dias são feitas duas reuniões no departamento de embarque e desembarque de materiais, onde são analisadas todas as demandas enviadas pelo pessoal da programação de embarcações.

Nessa reunião é definida toda a escala (associação de cada demanda a determinadas embarcações) de embarque e desembarque de produtos, sendo que, na reunião da manhã é definida a escala de prioridade das embarcações do dia e, na reunião da noite, é definida a escala de prioridade da noite, onde são dadas prioridades aos embarques de materiais de rancho.

Os pedidos feitos pelas plataformas são enviados à base especificando, entre outras coisas, a data e hora mínima (data mais cedo) em que pode receber o material e a data e limite (data mais tarde) para o recebimento. Isso é feito para que o caso de uma embarcação chegar à plataforma e não poder entregar o material solicitado pelo fato da plataforma estar em alguma operação que impossibilite o desembarque de material, ocorra somente quando o pessoal da plataforma se vê obrigado a realizar alguma operação de emergência não programada. Além disso, os prazos definidos nos pedidos são, também, levados em conta na definição da fila de atendimentos do dia. Ao se definir uma embarcação a atracar, considera-se a rota a ser realizada, a demanda para a rota, os períodos para atendimento e o tipo de embarcação. A ordem de chegada no porto é levada em consideração, na definição de prioridade de embarcação, somente após definidos todos os critérios anteriores.

Assim, tendo-se os pedidos que devem ser entregues no dia e as embarcações a chegarem, tem-se todas as informações necessárias para a definição da seqüência de atendimento nos berços.

Todas as embarcações que chegam ao porto de Imbetiba sempre trazem materiais das plataformas a serem desembarcados (com exceção das embarcações de emergência) que, em sua grande maioria, ficam nos barcos até que haja solicitação de carga para o barco. Dessa forma, há somente uma operação de atracação do barco no berço, tanto para embarque, quanto desembarque de materiais.

A exceção ocorre quando a embarcação traz algum material que não pode esperar até que haja carga para ser embarcada. Assim, esta atraca, desembarca o material, depois vai para a área de fundeio. A área de fundeio é a área onde ficam todas as embarcações que estão aguardando cargas, ou estão na fila aguardando para serem atendidas.

2.3. Entrega de materiais

Os expressinhos, que são um tipo de embarcação de emergência, porém, com horários pré-definidos de saída do porto, saem, geralmente, da seguinte forma: um, às 14:00h e 2 às 23:00h para entregas pequenas e de emergência nas plataformas. Esses barcos são programados para usar, normalmente, um determinado berço.

Quando se trata de uma entrega de emergência, que não pode esperar o horário pré-definido dos expressinhos de saída do porto, nem entrar na fila de programação, pode acontecer de uma embarcação ter que sair de um berço para dar lugar ao expressinho, ou qualquer outra embarcação que seja necessária para levar o material solicitado.

3 – ALGORITMOS PROPOSTOS PARA O PROBLEMA

O problema de sequenciamento das embarcações, para carga/descarga no porto, pode ser modelado de forma similar a um clássico problema da literatura científica denominado "Escalonamento de Tarefas em Máquinas Paralelas" [Armentano & Yamashita, 2000; Armentano & Toledo, 1997; Arroyo & Ribeiro, 2004; França Filho & Armentano, 2000; Yamashita *et al.*, 2000]. Neste problema, existem um conjunto de máquinas não necessariamente idênticas e um conjunto de operações ou tarefas a serem processadas nas máquinas. Cada operação deve ser processada em uma única máquina. Os tempos de processamento das operações nas máquinas são conhecidos. O problema consiste em determinar a ordem de processamento das operações nas máquinas minimizando o tempo total de processamento (*makespan*).

Associando ao problema em questão, as máquinas poderiam ser os berços juntamente com os equipamentos usados para fazer a carga/descarga de suprimentos (guindastes, empilhadeiras e carretas). Uma operação seria o conjunto de suprimentos a serem carregados numa embarcação. O objetivo é definir a ordem de atendimento das embarcações no porto, minimizando o atraso total. O atraso total é calculado somando o tempo de atraso de todas as embarcações, onde o tempo de atraso da embarcação i é calculado da seguinte forma: $(t_i - rt_i) \times pp_i$. t_i representa o momento em que a embarcação i foi atendida, rr_i representa o *release-time* da embarcação i, ou seja, o momento a partir do qual a embarcação i está liberada para atracar, e pp_i representa um peso dado de acordo com a prioridade da embarcação i. Quanto mais prioritária for uma embarcação, maior o peso desta, fazendo com que mais representativo seja seu tempo de espera.

Nesta seção serão descritos todos os algoritmos criados para o problema, desde uma breve descrição sobre a metaheurística GRASP, os algoritmos construtivos criados, as buscas locais e, por fim, os algoritmos GRASP gerados a partir das combinações dos métodos construtivos e de busca local criados.

3.1. A metaheurística GRASP

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 artigo são desenvolvidas heurísticas GRASP para resolver o problema de escalonamento de embarcações no porto de Imbetiba. O objetivo é definir qual destas heurísticas se adapta melhor ao problema em questão.

A Figura 1 apresenta o pseudocódigo padrão de uma heurística GRASP que recebe como parâmetro de entrada o número de iterações a serem realizadas (*N_iter*). Como saída, a heurística retorna a melhor solução encontrada, *Sol*. O laço nas linhas 1-7 realiza as *N_iter* iterações GRASP. A solução *s* é construída na linha 3 (o parâmetro α representa o grau de aleatoriedade na etapa de construção) e refinada na linha 4. Se a solução *s'*, gerada na etapa de busca local, for a melhor solução até o momento, esta é armazenada em *Sol*. Por fim, é retornada, na linha 8, a melhor solução encontrada, *Sol*.

3.2. Métodos construtivos

Nas subseções seguintes serão apresentados os algoritmos construtivos gerados para o problema em questão. Foram construídos três algoritmos considerando diferentes pontos importantes para cada método.

Procedimento GRASP (*N_iter*)
Entrada
N_iter - número de iterações GRASP.
Saída
Sol – melhor solução encontrada.
Início
01. **Para** *i* **de** 1 **até** *N_iter* **faça**
02. **Início**
03. $s \leftarrow$ **Construção** (α);
04. s' $\leftarrow$ **BuscaLocal** (s);
05. **Se** s' for a melhor solução até o momento **então**
06. *Sol* $\leftarrow$ s';
07. **Fim_para**
08. **Retorne** *Sol*;
Fim_GRASP

Figura 1. Algoritmo GRASP padrão.

3.2.1. Escalonamento de tarefas por prioridades (ETP)

O algoritmo de escalonamento de tarefas por prioridades considera que, quanto menor a prioridade dada a uma tarefa, mais importante é a tarefa e mais rápida esta deve ser executada. Desta forma, deve-se classificar as tarefas a serem executadas em ordem crescente de prioridade, de forma que as tarefas de menor grau de prioridade sejam atendidas sempre primeiro que as de maior grau.

Quando se tem mais de uma tarefa com o mesmo grau de prioridade, classifica-se, em ordem crescente, pelo *release time* da mesma. A partir das tarefas classificadas, gera-se a solução inicial incluindo-se cada tarefa em uma das máquinas que podem atendê-la, respeitando sua classificação na fila de 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 α primeiras tarefas da fila de prioridades, onde α é um parâmetro de entrada do algoritmo.

Quando uma tarefa pode ser executada por mais de uma máquina, escolhe-se aleatoriamente a máquina a executar a tarefa.

3.2.2. Escalonamento de tarefas por quantidade de máquinas que podem executá-las, onde a posição inicial esteja livre (ETQM1)

Nesta estratégia construtiva foi considerada, primeiramente, a quantidade de máquinas que podem executar determinada tarefa, tomando-se como fator decisório as tarefas que têm menos opções de máquinas para executá-las. O intuito desta estratégia é diminuir o tempo de espera, uma vez que as tarefas com um menor número de possíveis máquinas a atender 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, a ordem de classificação será o grau de prioridade das tarefas (considerando-se a ordem crescente das prioridades, conforme descrito na Subseção 3.2.1).

Assim como no algoritmo anterior, 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 α primeiras tarefas, onde α é um parâmetro de entrada do algoritmo.

Ao se inserir uma tarefa na máquina, deve-se escolher a melhor posição considerando todas as posições disponíveis na máquina, a partir do *release time* da tarefa, de forma que o atraso total (soma do atraso de todas as tarefas alocadas à maquina) na máquina seja o menor possível. Quando uma tarefa puder ser executada por mais de uma máquina, escolhe-se a máquina onde a tarefa não ocasionar atraso ou ocasionar o menor atraso possível.

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

3.2.3. Escalonamento de tarefas por quantidade de máquinas que podem executá-las, onde, se a posição inicial estiver ocupada, considera-se a de melhor prioridade (ETQM2)

Para esse tipo de escalonamento foram considerados, basicamente, os mesmos critérios do escalonamento anterior. A diferença é que, se a posição relativa ao *release time* da tarefa já estiver ocupada por outra tarefa na máquina, considera-se dentre estas duas tarefas a de melhor prioridade. Dessa forma, se a tarefa a entrar na máquina tiver prioridade melhor que a que já está na máquina, esta última é empurrada para frente e a primeira entra antes dela. Caso contrário, a tarefa a ser inserida será colocada na primeira posição após a tarefa que está na máquina.

3.3. Buscas locais

Nas subseções seguintes serão apresentados dois algoritmos de busca local propostos para o problema em questão.

3.3.1. Busca local BL1

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 berços diferentes. Para que a troca seja analisada neste tipo de busca local, primeiramente é verificado se as embarcações estão alocadas em berços diferentes. Depois, verifica-se se ambas as embarcações podem trocar de berços, ou seja, para que a troca da embarcação *a*, que está no berço *X*, pela

embarcação *b*, que está no berço *Y* possa ser analisada, é necessário que a embarcação *a* possa ser alocada no berço *Y* e a embarcação *b* possa ser alocada no berço *X*. Se esta condição for satisfeita, a troca dos barcos é analisada e um novo atraso é calculado nos berços *X* e *Y*. Se os resultados obtidos forem melhores que os atuais nos berços, realiza-se a troca e a busca local continua até que todas as trocas possíveis sejam analisadas e nenhuma delas melhore a solução atual (tenha um atraso total menor do que o da solução atual).

3.3.2. Busca Local BL2

Este procedimento de busca local é equivalente ao procedimento de busca local BL1, diferindo, apenas, no fato que serão consideradas somente as trocas entre embarcações que se encontram no mesmo berço.

3.4. Algoritmos GRASP desenvolvidos

Como citado anteriormente, o algoritmo GRASP padrão tem a estrutura apresentada na Figura 1, onde, durante um certo número de iterações, uma solução é criada por um método construtivo e depois refinada por um procedimento de busca local. Os algoritmos GRASP que serão apresentados nesta subseção são combinações entre os métodos construtivos propostos (ver Subseção 3.2) e os procedimentos de busca local propostos (ver Subseção 3.3). Desta forma, os métodos GRASP gerados foram os seguintes:

i. Algoritmo **GRASP1** – utiliza o método construtivo **ETP** descrito na Subseção 3.2.1 e o procedimento de busca local **BL1** descrito na Subseção 3.3.1.
ii. Algoritmo **GRASP2** – utiliza o método construtivo **ETQM1** descrito na Subseção 3.2.2 e o procedimento de busca local **BL1** descrito na Subseção 3.3.1.
iii. Algoritmo **GRASP3** – utiliza o método construtivo **ETQM2** descrito na Subseção 3.2.3 e o procedimento de busca local **BL1** descrito na Subseção 3.3.1.
iv. Algoritmo **GRASP4** – utiliza o método construtivo **ETP** descrito na Subseção 3.2.1 e o procedimento de busca local **BL2** descrito na Subseção 3.3.2.
v. Algoritmo **GRASP5** – utiliza o método construtivo **ETQM1** descrito na Subseção 3.2.2 e o procedimento de busca local **BL2** descrito na Subseção 3.3.2.
vi. Algoritmo **GRASP6** – utiliza o método construtivo **ETQM2** descrito na Subseção 3.2.3 e o procedimento de busca local **BL2** descrito na Subseção 3.3.2.

4. Resultados computacionais

Todos os experimentos computacionais foram feitos em um processador Atlhon 3200 e 520Mb de memória RAM. Os algoritmos GRASP aqui propostos foram implementados em C utilizando a versão 6.0 do compilador Microsoft Visual C++.

Para realizar uma comparação entre os algoritmos GRASP propostos foram gerados 16 problemas testes, que possuem de *nb* = 5 a 8 berços 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 berços do problema (*nb*) e o tempo total para escalonamento (T).

Tabela 1 – Problemas testes.

Problema teste	**Número de berços (*nb*)**	**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

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 (*te*) foi definido aleatoriamente entre 1 e 24 horas. A quantidade de embarcações do problema foi definido pela seguinte fórmula: (*nb***T*)/12, onde o denominador representa o tempo médio de atendimento (embarque/desembarque) de uma embarcação. A prioridade de atendimento de cada embarcação foi definida aleatoriamente entre os valores de 0 a 5. Os pesos associados a cada prioridade (*pp*) foi o seguinte: a prioridade 0 (mais prioritária) tem peso 6, a prioridade 1 tem peso 5, a prioridade 2 tem peso 4 e assim sucessivamente.

No experimento realizado, cada algoritmo GRASP proposto foi executado 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 em C do gerador descrito em [Schrage, 1979].

Tabela 2 – Resultados do primeiro experimento.

	GRASP1		GRASP2		GRASP3		GRASP4		GRASP5		GRASP6	
	Atraso	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**
Ins5-48	5098,0	8,1	367,6	76,1	396,2	22,1	5962,0	2,6	**292,8**	67,4	394,4	12,5
Ins5-72	2917,6	15,2	696,0	111,6	754,0	42,2	3601,4	5,3	**463,6**	102,1	689,4	20,3
Ins5-96	1877,0	72,6	1178,0	219,7	1222,6	133,7	2362,8	11,8	**1006,2**	141,6	1174,8	35,2
Ins5-120	2494,0	109,4	1600,6	400,9	1661,6	189,2	3048,6	21,6	**1517,2**	278,9	1749,6	51,06
Ins6-48	2383,0	14,9	454,2	121,9	538,2	42,7	3566,0	3,7	**381,4**	273,2	499,4	21,2
Ins6-72	3631,0	49,9	1171,6	192,8	1214,4	78,26	4498,2	9,72	**710,6**	178,7	1212,4	33,4
Ins6-96	3489,6	99,5	1706,8	405,8	1885,6	180,22	3784,2	20,4	**1205,4**	327,4	1910,6	56,9
Ins6-120	3353,4	158,0	1010,0	622,6	1343,2	277,8	3419,6	37,8	**874,8**	400,2	1345,8	81,3
Ins7-48	1060,2	25,3	584,0	159,6	647,0	53,9	1073,0	5,0	**500,0**	139,1	675,0	30,2
Ins7-72	4918,4	68,7	1569,8	290,8	**1306,2**	126,7	5352,0	13,5	1392,6	232,4	1353,8	54,8
Ins7-96	5136,2	199,0	2237,8	569,8	2403,6	333,6	6134,2	30,3	**2172,8**	361,9	2391,4	94,1
Ins7-120	5605,8	281,6	2244,0	972,8	2241,6	458,7	7949,4	53,3	**1816,6**	583,4	2183,8	135,8
Ins8-48	1201,6	25,9	**802,2**	197,5	1053,6	64,4	1116,2	7,3	927,2	211,8	1040,0	38,3

Ins8-72	3483,0	115,1	1891,8	459,8	2420,0	181,7	3628,0	20,7	**1757,2**	360,3	2416,0	78,8
Ins8-96	4099,4	205,5	1770,0	795,7	2015,6	333,1	4336,6	44,8	**1639,8**	1406,6	2128,8	122,4
Ins8-120	5039,6	354,1	2063,2	1287,9	2419,2	622,5	5633,4	76,9	**1953,4**	946,4	2420,8	179,5

A Tabela 2 apresenta os custos (atraso total) e os tempos (em segundos) obtidos, para cada problema teste, por cada algoritmo GRASP proposto. Lembrando que o atraso total é calculado somando o tempo de atraso de todas as embarcações, onde o tempo de atraso da embarcação *i* é calculado da seguinte forma: $(t_i - rt_i) \times pp_i$. t_i representa o momento em que a embarcação *i* foi atendida, rr_i representa o *release-time* da embarcação *i*, ou seja, o momento a partir do qual a embarcação *i* está liberada para atracar, e pp_i representa um peso dado de acordo com a prioridade da embarcação *i*.

Na Tabela 2 foi destacado, em negrito, os melhores atrasos obtidos para cada problema teste. O algoritmo GRASP5 obteve o melhor resultado para 14 dentre os 16 problemas testes. Para o problema teste "Ins7-72", o algoritmo GRASP3 obteve o melhor resultado. Para o "Ins8-48", o algoritmo GRASP2 obteve o melhor resultado. Com base neste experimento, o algoritmo GRASP5 se mostra como o mais adaptado ao problema tratado neste trabalho.

5. Conclusão

Neste trabalho foi abordado o problema de minimizar o tempo de espera das embarcações no porto de Imbetiba. 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. Sendo assim, para efeito de comparação, foi necessário desenvolver diferentes algoritmos com o intuito de escolher entre eles o que mais se adapta ao problema em questão.

O problema abordado é uma variação do problema de escalonamento de tarefas em máquinas paralelas [Armentano & Yamashita, 2000; Armentano & Toledo, 1997; Arroyo & Ribeiro, 2004; França Filho & Armentano, 2000; Yamashita *et al.*, 2000], 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]. Foram desenvolvidos, então, seis algoritmos baseados na metaheurística GRASP (ver Seção 3). Dentre eles o que se mostrou mais adequado ao problema em questão foi o algoritmo GRASP5, que utiliza na sua etapa de construção o método **ETQM1** descrito na Subseção 3.2.2 e na etapa de busca local o procedimento **BL2** descrito na Subseção 3.3.2., uma busca local mais simples, quando comparada coma a busca local **BL1** descrita na Subseção 3.3.1, mas que mostrou melhor desempenho quando combinada com o método **ETQM1.**

Referências bibliográficas

1. ARMENTANO, V. A., YAMASHITA, D. S. Tabu Search for Schedulling on Identical Parallel Machines to Minimizing Mean Tardiness. *Journal of Intelligent Manufacturing*, Vol.11, No.5, p.453- 460, 2000.
2. ARMENTANO, V. A., TOLEDO, F. M. B. Dynamic Programming Algorithms for the Parallel Machine Lot Sizing Problem. *Pesquisa Operacional*, Brasil, v. 17, p. 137-149, 1997.
3. 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.

4. ARROYO, J. E. C.; VIEIRA, P. S. e VIANNA, D. S. A GRASP algorithm for the multi-criteria minimum spanning tree problem, *Multidisciplinary Conference on Scheduling: Theory and Applications*, Nova York, p.1-11, 2005.
5. CE_UFRJ. Descrição de análise do processo de movimentação de materiais do porto de Imbetiba. *Centro de estudos da UFRJ*, 1997.
6. CE_UFRJ. Modelo de Simulação de Operações Portuárias: Descrição, Utilização e Análise. *Centro de estudos da UFRJ*, 1997.
7. 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*, 2001.
8. FEO, T.A., RESENDE, M.G.C. Greedy randomized adaptive search procedure. *Journal of Global Optimization*, v. 6, p. 109 – 133, 1995.
9. FRANCA FILHO, M. F., ARMENTANO, V. A. Restrição de Vizinhança em Busca Tabu: Uma Aplicação à Programação de Tarefas em Máquinas Paralelas. *In: XXXII Simpósio Brasileiro de Pesquisa Operacional*, Viçosa, p.734-746, 2000.
10. 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*, v.14, p.285 - 292, 1998.
11. PETROBRAS. Página oficial da Petrobras. *www.petrobras.com.br*, 2006.
12. RESENDE, M.G.C., e RIBEIRO, C.C. Greedy randomized adaptive search procedures. Em F. Glover e G. Kochenberger (eds.), *Handbook of Metaheuristics*. Kluwer, 219-249, 2003.
13. RIBEIRO, C. C. e VIANNA, D. S. A GRASP/VND heuristic for the phylogeny problem using a new neighborhood structure, *International transaction on operation research*, v.12, p.325-338, 2005.
14. SCHRAGE, L. A more portable FORTRAN random number generator, *ACM Transactions on Mathematical Software*, 5, 132-138, 1979.
15. 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*, v.1388, p.216 - 225, 1999.
16. YAMASHITA, D. S., MAZZINI, R., ARMENTANO, V. A. Busca Tabu Aplicada à Programação de Tarefas em Máquinas Paralelas com Tempos de Preparação Dependentes da Seqüência e Datas de Entrega. *In: XXXII Simpósio Brasileiro de Pesquisa Operacional*, Viçosa, p.389-396, 2000.