UNIVERSIDADE CANDIDO MENDES – UCAM
PROGRAMA DE PÓS-GRADUAÇÃO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

Sandra Regina Coelho

HEURÍSTICAS GRASP PARA UM PROBLEMA DE ESCALONAMENTO DE TAREFAS EM MÁQUINAS PARALELAS EM TEMPO REAL BASEADO EM MECANISMO DE PRIORIDADES

Campos dos Goytacazes, RJ
Abril de 2008

FICHA CATALOGRÁFICA

Preparada pela Biblioteca da **UCAM - CAMPOS** 009/2009

Coelho, Sandra Regina.

Heurísticas GRASP para um problema de escalonamento de tarefas em máquinas paralelas em tempo real baseado em mecanismo de prioridades / Sandra Regina Coelho. - 2009.
56 f. ; il.

Orientador: Dalessandro Soares Vianna.
Dissertação de Mestrado em Pesquisa Operacional e Inteligência Computacional – Universidade Candido Mendes - Campos. Campos dos Goytacazes, RJ, 2008.
Bibliografia: f. 54 - 56.

1. Programa Meta heurístico 2. GRASP (computer file), VND 3. Escalonamento de tarefas I. Universidade Candido Mendes – Campos. II. Título.

CDU – 004.023

UNIVERSIDADE CANDIDO MENDES – UCAM
PROGRAMA DE PÓS-GRADUAÇÃO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

Sandra Regina Coelho

HEURÍSTICAS GRASP PARA UM PROBLEMA DE ESCALONAMENTO DE TAREFAS EM MÁQUINAS PARALELAS EM TEMPO REAL BASEADO EM MECANISMO DE PRIORIDADES

Dissertação apresentada ao Programa de Pós-Graduação em Pesquisa Operacional e Inteligência Computacional da Universidade Candido Mendes – Campos do Goytacazes/RJ, para a obtenção do grau de MESTRE EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

Orientador: Prof. DSc. Dalessandro Soares Vianna

Campos dos Goytacazes, RJ
Abril de 2009

Às minhas filhas, que souberam entender e me apoiaram sempre que precisei, com todo amor, humildade e compreensão na pureza de seus corações simples de crianças.

Ao meu irmão Marcos, pela solidariedade e amor me apoiando sempre no que estava ao seu alcance.

Agradecimentos

À Deus, que me concedeu forças, ânimo e sabedoria para começar e seguir até o fim toda essa jornada.

Ao meu orientador, Dalessandro, pela amizade, compreensão e ajuda, sem o qual nada desse trabalho teria sido realizado e me deu todo o apoio necessário em especial quando tive que terminar o trabalho à distância.

Aos amigos de mestrado, e, em especial, a Liamara, por toda a amizade e apoio despendidos sempre que precisei.

Aos profissionais da PETROBRAS, que forneceram as informações fundamentais para a montagem do problema tema deste trabalho.

À FAPERJ, ao CNPQ e à FENORTE/TECNORTE pelo apoio financeiro recebido.

À Prefeitura Municipal de Campos dos Goytacazes, pela bolsa concedida durante o meu primeiro ano de estudo.

Aos membros da banca examinadora, por aceitar o convite e pela contribuição com as sugestões para a melhoria desse trabalho.

Aos meus pais e irmãos, pelo carinho, apoio e incentivo constante.

A todos aqueles que, de alguma forma, contribuíram para a conclusão desse trabalho.

RESUMO

Petróleo Brasileiro S/A (PETROBRAS) iniciou suas atividades no século XX e é, atualmente, a maior companhia brasileira no ramo de energia. Ela atua na exploração, produção, refino, comercialização e transporte de derivados de petróleo no Brasil e em outros países. A maioria do petróleo produzido está concentrada na bacia de Campos, onde está localizado o porto da empresa (Porto de Imbetiba – Macaé/RJ). Todas as plataformas são abastecidas utilizando este porto.

Pesquisas feitas no porto de Imbetiba mostram a necessidade de se otimizar, entre outros, o problema do tempo de espera das embarcações. Neste problema, a ordem de atendimento das embarcações deve ser decidida e, de acordo com o material que cada uma carrega, as seguintes restrições devem ser respeitadas: cada embarcação possui um conjunto de berços que podem atendê-la; cada uma tem uma prioridade de atendimento e o tempo mínimo em que pode ser atendida, a qual, antes deste tempo, a embarcação não pode ser atendida.

Este problema pode ser associado ao “Problema de Escalonamento de Tarefas em Máquinas Paralelas”, onde um subconjunto de tarefas, cada uma com um tempo de execução, deve ser organizado em um conjunto de máquinas, de forma que o tempo total de execução seja minimizado. No problema estudado, as máquinas são os berços e as tarefas são as embarcações, adicionando a restrição de prioridade, o conjunto de máquinas que podem atender uma tarefa e o tempo mínimo de espera de uma tarefa deve ser considerado.

Este trabalho propõe diferentes algoritmos GRASP, onde alguns usam a tecnologia VND como busca local. Três estruturas de vizinhanças são implementadas: Troca, Intercâmbio e Realocação. Os resultados computacionais mostram a eficiência dos métodos propostos.

PLAVRAS-CHAVE: Metaheurística Grasp, VND, Otimização Combinatória, Problema de Escalonamento de Tarefas em Máquinas Paralelas

ABSTRACT

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 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 cradle 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 can not be attended.

This problem can be associated to the “Job Scheduling Problem in Parallel Machines”, where a set of jobs, each one with an execution time, must be organized on a set of machines, such as the total execution time is minimized. In the studied problem, the machines are the cradle, the jobs are the towboats and, besides, the additional restrictions of priority, subset of machines that can attend a job and the minimal time for a job scheduling, must be considered.

This work proposes diferents GRASP algorithms, where some of them uses the technique VND as local search. Three neighborhood structures were implemented: Exchange, Interchange and Relocation. The computational results show the efficiency of the proposed methods.

KEYWORDS: Metaheuristic GRASP; VND; Combinatorial Optimization; Job Scheduling Problem in Parallel Machines.

SUMÁRIO

1 – INTRODUÇÃO 11
2 – BREVE DESCRIÇÃO DO PROCESSO DE MOVIMENTAÇÃO DE MATERIAIS NO PORTO DE MACAÉ 13
3 – DESCRIÇÃO DAS RESTRIÇÕES DO PROBLEMA APRESENTADO 16
3.1 DEFINIÇÃO DOS TIPOS DE EMBARCAÇÕES, RECURSOS EXISTENTES E CLASSES DE MATERIAIS A SEREM TRANSPORTADOS 16
3.1.1 Embarcações 16
3.1.2 Recursos 17
3.1.2.1 Guindastes 17
3.1.2.2 Empilhadeiras 17
3.1.2.3 Berços 17
3.1.3 Classes de materiais 18
3.1.3.1 Suprimento de fluidos 18
3.1.3.2 Suprimento de convés 19
3.1.3.2.1 Suprimento de convés/DMM 19
3.1.3.2.2 Suprimento de rancho 19
3.1.3.2.3 Suprimento de tubos 19
3.1.3.2.4 Suprimento de granéis 20
3.2 DISTRIBUIÇÃO DO PORTO 20
3.3 DESCRIÇÃO DO FUNCIONAMENTO DOS PIERS 21
3.3.1 Funcionamento do píer 21
3.3.2 Abastecimento de fluidos 21
3.3.3 Granel 21
3.4 EMBARQUE E DESEMBARQUE DE MATERIAIS 22
3.4.1 Desembarque 22
3.4.2 Embarque 23
3.5 COMO SÃO DEFINIDOS OS ATENDIMENTOS ATUALMENTE 23
3.6 ENTREGA DE MATERIAIS 24
3.7 RESTRIÇÕES TRATADAS NESTE TRABALHO 25
4 – O PROBLEMA DE ESCALONAMENTO DE TAREFAS 27
4.1 ESCALONAMENTO DE TAREFAS EM MAQUINAS PARALELAS 28
4.2 ESCALONAMENTO EM TEMPO REAL 28
4.3 ESCALONAMENTO POR PRIORIDADES 30
4.4 ESCALONAMENTO USADO NO TRABALHO 30
5 – MÉTODOS CONSTRUTIVOS E DE BUSCA LOCAL GERADOS PARA O PROBLEMA 31
5.1 MÉTODOS CONSTRUTIVOS 31

5.1.1 Escalonamento de tarefas por prioridades (ETP) 31
5.1.2 Escalonamento de tarefas por quantidade de máquinas que podem executá-las, onde a posição inicial esteja livre (ETQM1) 32
5.1.3 Escalonamento de tarefas por quantidade de máquinas que podem executá-las, onde, se a posição inicial esteja ocupada, considere a tarefa de melhor prioridade (ETQM2) 35
5.2 ALGORITMOS DE BUSCA LOCAL 36
5.2.1 Busca local intercambio(BLI) 36
5.2.2 Busca local troca (BLT) 37
5.2.3 Busca local realocação (BLR) 38
6 – A METAHEURÍSTICA GRASP 40
6.1 ALGORITMOS GRASP 41
6.1.1 Resultados obtidos com as heurísticas GRASP 42
7 – ESTRATÉGIAS GRASP + VND PROPOSTAS 46
7.1 HEURÍSTICAS USANDO O ALGORITMO CONSTRUTIVO ETQM1 + VND (HVND) 47
7.2 RESULTADOS OBTIDOS COM AS HEURÍSTICAS USANDO O MÉTODO GRASP + VND 49
8 – CONCLUSÃO E TRABALHOS FUTUROS 52
9 – REFERÊNCIAS BIBLIOGRÁFICAS 54

LISTA DE FIGURAS

Figura 1 – Imagem da distribuição do Porto de Imbetiba localizado em Macaé 16
Figura 2 – Imagem de embarcação sendo abastecida com fluido ao mesmo tempo em que uma outra está embarcando/desembarcando materiais 19
Figura 3 – Algoritmo ETP desenvolvido 35
Figura 4 – Algoritmo ETQM1 desenvolvido 37
Figura 5 – Algoritmo ETQM2 desenvolvido 38
Figura 6 – Algoritmo BLI desenvolvido 40
Figura 7 – Algoritmo BLT desenvolvido 41
Figura 8 – Algoritmo BLR desenvolvido 42
Figura 9 –. Algoritmo GRASP desenvolvido. 44
Figura 10 – Algoritmo VND padrão. 50

1 – INTRODUÇÃO

A operação de suprimentos usando embarcações requer grandes investimentos e possui custos operacionais muito altos, principalmente no que se refere às diárias pagas às embarcações para o transporte dos mesmos e nas operações realizadas nos portos.

Por esse motivo, o tempo de espera das embarcações para embarque e desembarque de produtos nos portos pode representar uma diferença no custo final considerável às empresas que custeiam o transporte de produtos recebidos/enviados via embarcações. Visando esse problema, têm surgido alguns estudos para se tentar diminuir ao máximo esse tempo de espera, aumentando-se a eficiência dos serviços portuários.

Um modelo de simulação de operações portuárias foi apresentado a Petrobras em 1997, cujo principal objetivo foi possibilitar a modelagem do porto existente no sistema Petrobras em Macaé. O modelo apresentado, foi usado por duas vezes pela empresa, obtendo-se resultados que auxiliaram na tomada de decisão do procedimento a ser tomado para se melhorar o tempo de espera das embarcações no porto de Imbetiba, em Macaé.

Porém, como o modelo foi projetado objetivando-se a simulação das operações portuárias, para se decidir saber qual ação deve ser tomada para que o objetivo desejado seja alcançado, devem-se montar várias situações e, a partir da melhor simulação, decidir o que fazer. Ou seja, o usuário usa o método de “tentativa e erro” alterando os valores na entrada de dados (nos parâmetros de entrada), simula como ficariam as operações portuárias usando o sistema hoje existente na empresa e, de acordo com o resultado obtido, faz uma nova simulação com outros valores.

O trabalho proposto tem por objetivo desenvolver algoritmos para apresentar uma otimização do tempo em que as embarcações ficam paradas no porto, o que pode

representar a economia de um montante altíssimo para a empresa.

Diferente do modelo de simulação de operações portuárias apresentado, o foco desse trabalho está em se apresentar as melhores formas de atendimento das embarcações no porto de Imbetiba.

Os parâmetros de entrada são a quantidade de berços existentes no porto, a quantidade de embarcações existentes na fila de espera, a hora em que uma embarcação deve ser atendida (*release time*), a prioridade de atendimento de cada uma, o tempo de embarque/desembarque, a quantidade e quais berços podem atendê-la.

Os resultados apresentam as melhores ordens de atendimento da fila de espera em cada berço.

Esse trabalho irá descrever no capítulo 2 como funcionam as operações portuárias tomando-se, como referência, o porto da Petrobras na praia de Imbetiba em Macaé.

No capítulo 3 serão descritos os dados básicos e restrições que serão usados neste trabalho.

Os algoritmos de escalonamento de tarefas mais conhecidos serão apresentados no capítulo 4.

No capítulo 5 são descritos os métodos construtivos criados para este trabalho, as buscas locais criadas para o problema em questão.

Os métodos GRASP gerados e os resultados alcançados com os métodos GRASP estão descritos no capítulo 6..

No capítulo 7 são apresentados os métodos VND's criados para se conseguir melhores resultados e os resultados destes métodos criados.

No capítulo 8 são feitas comparações e comentários entre os melhores resultados obtidos.

2 – BREVE DESCRIÇÃO DO PROCESSO DE MOVIMENTAÇÃO DE MATERIAIS NO PORTO DE MACAÉ

Este capítulo tem como objetivo mostrar a abordagem utilizada para modelar as operações portuárias realizadas no porto de Macaé.

Parte dos estudos para esse trabalho foi baseada nas documentações [1,2] apresentadas para a construção do modelo de simulação de operações portuárias já existente na Petrobras hoje, adaptando-se aos resultados obtidos às mudanças realizadas na empresa após o levantamento feito anteriormente.

A análise das embarcações baseou-se no sistema de transporte marítmo e na situação portuária hoje existente. A partir destas análises, as embarcações foram agregadas em três tipos distintos: Suprimentos, Graneleiros e Exclusivas (ou especiais). Essas embarcações serão descritas no capítulo seguinte.

Os guindastes são usados para o embarque ou desembarque de materiais das embarcações e não saem do porto. Estes podem também ser usados para a carga ou descarga de materiais nas carretas, quando se tratar de materiais os quais as empilhadeiras não são capazes de transportar.

Para a carga e descarga de materiais das carretas, são usadas as empilhadeiras, que podem, também, movimentar materiais dentro do próprio porto de um setor para outro (pré-embarque de convés/pré-embarque de tubos, retroporto/área de desembarque de tubos, etc.).

As embarcações de suprimento podem ocupar qualquer um dos seis berços (Figura 1), 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 2 e 3, pois são os

únicos com 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 nos berços 5 ou 6 quando se trata de embarque/desembarque de materiais de ancoragem (âncora, estaca, torpedo, etc.), que necessitam de guindastes especiais.

Existem 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 (uma vez que determinadas embarcações só podem atracar em alguns berços específicos, como já citado).

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 se permite utilizar os estudos já realizados em outros portos, vistos que estes obedecem à regras padrões dos grandes portos, como, por exemplo, considerar a ordem de chegada das embarcações ou multa por atraso no atendimento, o que não é levado em conta no porto de Imbetiba.

Na Figura 1 é apresentada uma foto do porto de Imbetiba com os piers (de 1 a 3) e berços (de 1 a 6). Cada pier possui dois berços, chamados de lado praia e lado mar.

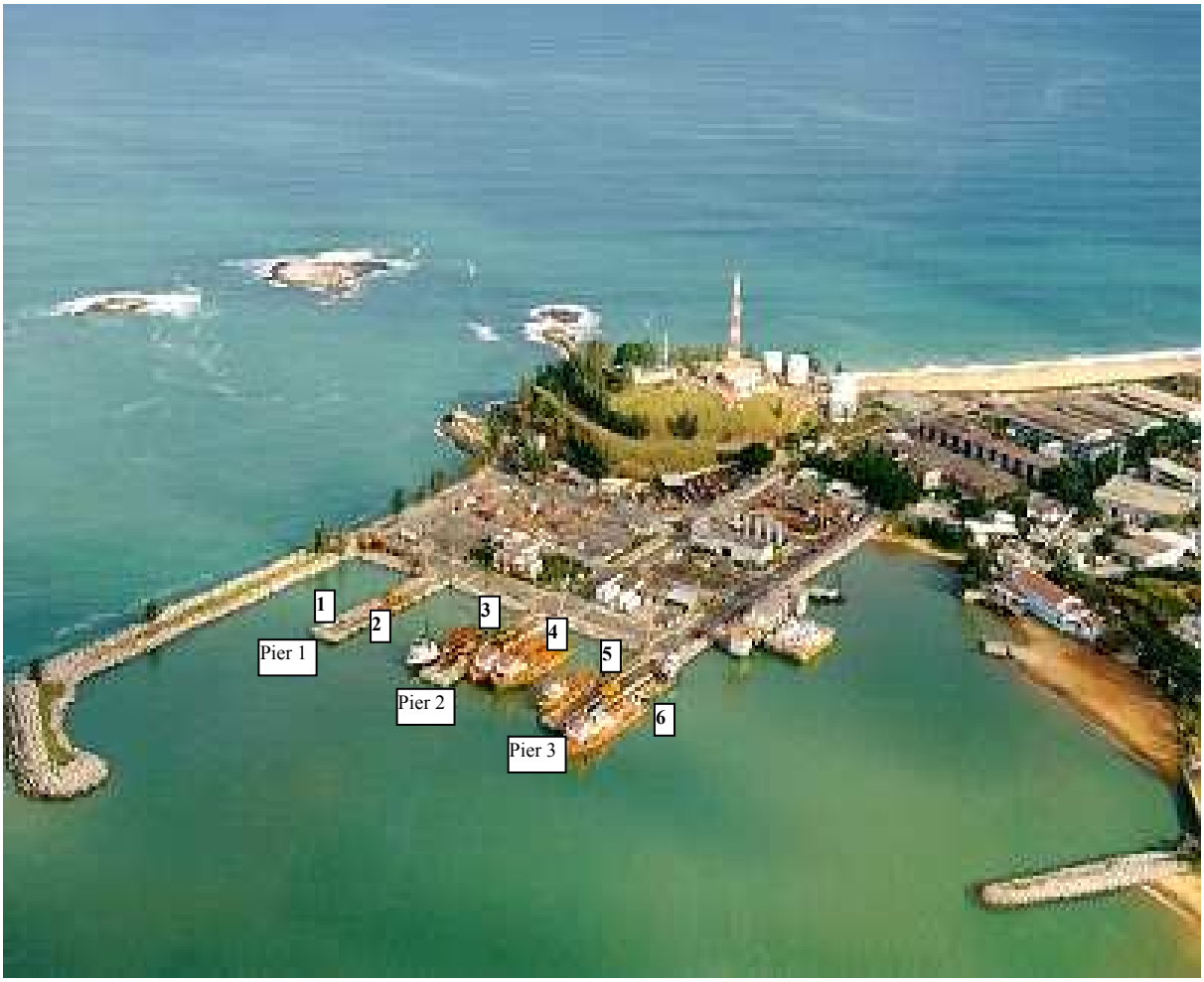

Figura 1 – Imagem da distribuição do Porto de Imbetiba localizado em Macaé.
Fonte: Fotos fornecidas pela Petrobras.

3 – DESCRIÇÃO DAS RESTRIÇÕES DO PROBLEMA APRESENTADO

Iniciando os estudos do trabalho foi feito um levantamento do funcionamento das operações portuárias no porto de Imbetiba em Macaé, os recursos utilizados, materiais embarcados e desembarcados e as operações que devem ser executadas para o embarque e desembarque de materiais das embarcações.

A seguir será descrito cada processo usado como ponto de partida para o entendimento e estudo do problema apresentado neste trabalho.

3.1 DEFINIÇÃO DOS TIPOS DE EMBARCAÇÕES, RECURSOS EXISTENTES E CLASSES DE MATERIAIS A SEREM TRANSPORTADOS.

3.1.1 Embarcações

- 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 de materiais do tipo granel, tais como cimento, baritina e bentonita. O carregamento é feito utilizando-se mangotes (tipo de mangueiras de dimensões grandes) e, eventualmente, algumas, utilizam guindastes para carregamento de materiais.

- Exclusivas – é o tipo de embarcação que fica mais tempo no píer. 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.

3.1.2 Recursos

3.1.2.1 Guindastes

Podem, também, ser chamados de servidores e são usados para fazer o manuseio de cargas das carretas para as embarcações, ou vice-versa. Cada berço possui o seu guindaste exclusivo.

3.1.2.2 Empilhadeiras

As empilhadeiras são usadas para a retirada de material das carretas.

3.1.2.3 Berços

Os berços são os locais onde as embarcações ficam fazendo operações com guindastes, ou abastecimento de água, óleo e lama.

Se uma embarcação for realizar o abastecimento de fluidos, esta pode atracar junto a uma outra embarcação em um berço, uma vez que não depende de recursos do píer para seu abastecimento e sim dos mangotes. A Figura 2, apresenta a foto de duas embarcações num mesmo berço, onde uma está trabalhando com embarque/desembarque e a segunda sendo abastecida com fluidos.

Figura 2 – Imagem de embarcação sendo abastecida com fluido ao mesmo tempo em que uma outra está embarcando/desembarcando materiais.
Fonte: Foto fornecida pela Petrobras.

3.1.3 Classes de materiais

Os materiais estão divididos em classes visando padronizar as solicitações feitas pelos usuários do STM (sistema que controla a demanda de materiais no porto). No total, existem 21 classes, porém, neste trabalho, serão descritos somente os principais.

3.1.3.1 Suprimento de fluidos

Essa classe está subdividida em:

- suprimento de água
- suprimento de diesel
- suprimento de lama

O carregamento de água e diesel são feitos por mangotes específicos que conduzem a água e o diesel dos reservatórios até a embarcação. Estes tipos de

carregamentos são feitos por gravidade (os reservatórios ficam no ponto mais alto do porto, para aproveitar o desnível em relação ao pier) e, tão logo a embarcação chega ao píer, um mangote é conectado a ela para abastecê-la. Com isso, dificilmente estes tipos de classes de operação fazem com que atrase a saída de uma embarcação.

O carregamento de lama também é feito através de mangotes, porém, pela estação de fluidos localizada no próprio porto de Imbetiba, onde ocorre a mistura de materiais para formar a lama.

3.1.3.2 Suprimento de convés

Este é caracterizado como o que possui as mais variadas formas, ou seja, pode ser encontrado sob a forma de containers, cestas metálicas, skids (suporte, em geral de metal, para transporte de objetos pesados e de difícil movimentação – em geral grandes máquinas) e outras formas que são transportadas no convés da embarcação, com exceção de tubos, ranchos, e materiais para movimentação de plataforma.

3.1.3.2.1 Suprimento de convés/DMM

Esta classe é composta por amarras, âncoras, bóias e outros materiais utilizados na movimentação de plataformas flutuantes.

3.1.3.2.2 Suprimento de rancho

É composto por containers que, em sua grande maioria, é padronizado.

3.1.3.2.3 Suprimento de tubos

Esse tipo de suprimento caracteriza-se pela variedade de tubos existentes, especialmente no que diz respeito ao diâmetro. Neste tipo de suprimento existem alguns acessórios que dificultam a movimentação.

3.1.3.2.4 Suprimento de granéis

O suprimento de granéis são divididos em:

- Cimento
- Baritina
- Bentonita

Estes ficam armazenados na planta de granéis e seu carregamento também é feito através de mangotes específicos.

3.2 DISTRIBUIÇÃO DO PORTO

O porto hoje possui 6 berços, distribuídos em 3 pieres com capacidade para comportar até 12 embarcações pequenas, 6 grandes ou suas combinações. Essa quantidade pode ser superada no caso de haver embarcações de contrabordo, ou seja ao lado de embarcações atracadas (abastecendo fluidos, por exemplo)

A distribuição dos pieres está da seguinte forma:

- berço 1 – não atende embarcações exclusivas;
- berços 2 e 3 – são os únicos que atendem embarcações graneleiras;
- berços 5 e 6 – são os que possuem guindastes mais potentes e, portanto, são os únicos que podem embarcar materiais pesados.

Os silos pertencentes à planta de granéis estão situados próximos aos pieres e estão distribuídos da seguinte forma: 6 silos de cimento, 3 de baritina e 3 de bentonita. O bombeio é feito através de 2 bombas (cada uma com o seu mangote), uma para o cimento, outra para a baritina/bentonita.

A estação de fluidos localiza-se próxima à planta de graneis, porém um pouco mais afastada do píer.

O setor de retroporto atua no embarque, desembarque e no atendimento de usuários da rede. Toda a carga que chega para embarque é registrada e conferida nesse setor com a carga ainda na carreta que a transportou.

Após isso, a carga é transportada para a área de pré-embarque, para a área de armazenagem de embarque de tubos ou diretamente para o berço, no caso da

embarcação já se encontrar disponível com o programa de operações pronto ou se tratar de material de difícil manuseio.

Uma carga nunca é enviada à área de retroporto ou pré-embarque sem que ela já esteja com a data e horário de embarque pré-definidos. Isso é feito devido a falta de espaço no porto. Dessa forma, uma carga nunca fica parada no porto por muito tempo (sempre é embarcada no mesmo dia).

No caso de desembarque de material, o retroporto é responsável pela conferência da carga e auxílio no descarregamento da carreta na área do retroporto, na triagem e no embarque e desembarque de tubos.

O retroporto é responsável, também, pela entrega de todo o material para o usuário Petrobras e algumas contratadas, exceto no caso de containers de contratadas.

3.3 DESCRIÇÃO DO FUNCIONAMENTO DOS PIERS

3.3.1 Funcionamento do píer

O pier trabalha 24h por dia, 7 dias por semana.

O abastecimento de fluídos e granel não influem nos horários de atendimentos, pois necessitam apenas da colocação de mangote na embarcação.

3.3.2 Abastecimento de fluidos

O abastecimento é feito somente em embarcações do tipo suprimento ou graneleira.

3.3.3 Granel

O abastecimento de granel é acionado juntamente com o abastecimento de água e diesel e é dividido em 4 fluidos:

- Baritina – exclusivo de embarcações graneleiras;

- Bentonita - também exclusiva de embarcações graneleiras. O abastecimento da bentonita é feito usando-se o mesmo mangote da baritina;

- Cimento – abastecimento pode ser feito por embarcações de suprimento e carregamento;

- Lama – exclusivo de embarcações de suprimento e carregamento.

3.4 EMBARQUE E DESEMBARQUE DE MATERIAIS

3.4.1 Desembarque

A operação de desembarque é feita antes que qualquer operação de embarque aconteça, para que seja desocupada a área do barco para o embarque de materiais. Essa operação ocorre sempre que a embarcação traz material recolhido das plataformas para serem desembarcadas no porto (para reparo ou para depósito).

Assim que a embarcação atraca no berço, o respectivo guindaste é acionado para colocar o material na carreta. Depois disso, ela vai para o retroporto, onde é feita a checagem da carga e, então, escolhido o destino do material, que podem ser:

- área de retroporto – no caso de material de convés;
- área de desembarque de tubos – no caso de tubos;
- empresas contratadas – neste caso não utilizam carretas Petrobras.

No destino, a empilhadeira local é acionada para descarregar a carreta e, depois, liberá-la para ir ao píer pegar mais material ou para voltar ao pool de carretas.

Se uma embarcação tiver cargas para mandar para mais de uma área de destino, a quantidade de carretas necessárias é requisitada ao mesmo tempo, quando entrarão em uma fila no berço e aguardarão o carregamento.

Se não houver carreta suficiente, forma-se uma fila de requisições que será atendida por ordem de chegada (FIFO) assim que forem sendo liberadas.

3.4.2 Embarque

Nesta fase, serão usados os recursos do pier para o carregamento da embarcação.

As áreas de origem dos materiais são as mesmas de destino do desembarque.

As carretas são carregadas nessas áreas pelas respectivas empilhadeiras, vão para o berço de destino, onde será usado o guindaste do pier para carregar a embarcação.

Os embarques dos containers de rancho são feitos somente à noite, visto que, nesse horário, a temperatura ambiente está mais baixa que durante o dia. Dessa forma, o transporte pode ser feito com uma preocupação menor quanto ao fato do material se estragar.

3.5 COMO SÃO DEFINIDOS OS ATENDIMENTOS ATUALMENTE

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 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 não ocorra o fato de uma embarcação chegar à plataforma e não poder entregar o material solicitado pelo fato de a plataforma estar em alguma operação que impossibilite o desembarque de material; ou, 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 a prioridade de uma embarcação a atracar, considera-se a rota a ser realizada, o tipo de embarcação, o tipo de material a ser embarcado/desembarcado e se trata-se de uma embarcação de emergência ou não.

Assim, tendo-se os pedidos que devem ser entregues no dia e as embarcações a chegar, tem-se todas as informações necessárias para a definição do atendimento a ser realizado no dia.

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.

Um tempo de espera para atendimento das embarcações que entram na fila de fundeio é contado a partir do momento em que a embarcação entra na fila, até o momento em que é chamada para atracar no porto.

Durante o momento da atracação de uma embarcação no porto, nenhuma outra operação pode estar sendo realizada no barco, inclusive, abastecimento.

Por determinação da Petrobras, os barcos especiais (ou embarcações exclusivas) de pesquisa ficam, uma vez por mês, atracados no porto durante 24h para inspeção.

Os barcos de cronograma (embarcações especiais que realizam operações de ancoragem), geralmente, ficam 18h atracados no porto para o embarque e desembarque de materiais especiais e sempre saem às 0h.

3.6 ENTREGA DE MATERIAIS

Nas entregas de materiais às plataformas, o material de rancho têm prioridade de entrega sobre qualquer outro material que estiver sendo transportado. Dessa forma, uma embarcação que estiver transportando esse tipo de material fará a entrega de todos os

produtos de rancho primeiro e, somente depois de todos entregues, são feitas, então, as entregas do restante dos materiais.

Dessa forma, uma embarcação pode passar mais de uma vez em uma plataforma em uma mesma rota de entrega, porém, neste trabalho não serão consideradas as rotas das embarcações, nem a prioridade das cargas, apenas as prioridades das embarcações.

Os expressinhos são um tipo de embarcação de emergência, porém, com horários pré-definidos de saída do porto para entregas pequenas e de emergência nas plataformas. Esses barcos são programados para usar, normalmente, o berço 1.

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.

3.7 RESTRIÇÕES TRATADAS NESTE TRABALHO

Neste trabalho foi definido como objetivo a ordem de atendimento das embarcações, minimizando o tempo de espera total e levando em conta a prioridade das embarcações.

Também foi considerado aqui que uma embarcação atraca uma única vez no berço e um berço atende apenas uma embarcação de cada vez, sendo esse atendimento realizado até o fim não permitindo interrupções.

Para avaliação dos algoritmos desenvolvidos neste trabalho foram gerados 16 problemas testes da seguinte forma:

Os problemas testes foram gerados variando-se a quantidade de berços de 5 a 8 e o tempo máximo de trabalho em cada berço variando de 48h a 120h (48, 72, 96 e 120h). Dessa forma, os problemas testes foram representados da seguinte maneira:

Problema teste	Número de berço (*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

Tabela 1 – Problemas testes gerados.

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. O número de embarcações(*N_emb*) do problema foi definido pela seguinte fórmula: (*nb***T*)/13, 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 *nb*, considerando que 0 é a prioridade mais alta e *nb*, a mais baixa. Os pesos associados a cada prioridade (*pp*) foi o seguinte: a prioridade 0 são as embarcações que não podem atrasar (embarcações de emergência ou os expressinhos com hora marcada), portanto, tem peso 50, a prioridade 1 tem peso *nb* , a prioridade 2 tem peso *nb-1* e assim sucessivamente. O release time da tarefa é dado como rt_i e at_i o momento em que a tarefa *i* é iniciada. O tempo de espera de uma tarefa é dado como $te_i = at_i - rt_i$. Assim, o atraso total é calculado da seguinte forma:

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

4 – O PROBLEMA DE ESCALONAMENTO DE TAREFAS

Os problemas de escalonamento de tarefas são problemas de alocação de recursos para execução de atividades concorrentes. Esses problemas encontram-se intimamente ligados ao planejamento e programação da produção nas indústrias, por isso é conveniente que seja adotada a mesma terminologia utilizada na indústria, aonde tarefas são atividades a serem executadas em máquinas que são recursos a serem alocados. Segundo Moccellin [17], a programação refere-se à ordenação das tarefas a serem executadas, em uma ou diversas máquinas considerando-se uma base de tempo, ou seja, determinando-se, principalmente, as datas de início e fim de cada tarefa.

Os tipos de escalonamentos apresentados no decorrer deste capítulo mostraram-se os mais adequados para o problema em questão, visto que são classificados como os mais intimamente ligados ao planejamento e programação de tarefas. Estes problemas são matematicamente desafiadores e extremamente difíceis de resolver. A explosão combinatória decorrente da necessidade de se examinar as várias alternativas de escalonamento existentes nesses problemas torna difícil e custosa a obtenção de uma solução ótima ou a mais próxima dela. Devido a isso, além da busca pelo entendimento desses problemas tão complexos, também se buscam novos métodos para resolvê-los de maneira rápida, fornecendo soluções de boa qualidade para propósitos práticos [1].

A literatura científica apresenta algumas aplicações de estratégias em problemas tais como: escalonamento de tarefas em processadores heterogêneos com restrições de precedência [11], roteamento de veículos [13], atribuição quadrática e sua utilização no mapeamento de tarefas a processadores em sistemas
multi-processadores [4, 5, 6, 7, 12, 15], caixeiro viajante [8, 9], roteamento de veículos com janelas de tempo [17], escalonamento de tarefas [14] e mapeamento de tarefas de um algoritmo paralelo a um conjunto de processadores idênticos [10]. A seguir,

serão descritos os tipos de escalonamentos encontrados que mais se aproximam ao objetivo desse trabalho.

4.1 ESCALONAMENTO DE TAREFAS EM MÁQUINAS PARALELAS (ETPM)

De acordo com Mendes et al. [18], o ETPM 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 ETPM encontra-se: um modelo de programação linear inteira, proposto por Dearing e Henderson (1984) [19], para a alocação do tear em uma tecelagem; um método heurístico, proposto por Sumicharst and Baker [20], baseado na solução de uma série de subproblemas de programação inteira 0-1 que melhora 2 resultados de Dearing e Henderson [19]; uma heurística busca tabu é proposta por França *et al.* [21]; Arroyo e Ribeiro [22] propõem um algoritmo genético para o ETPM com múltiplos objetivos; e uma heurística buca tabu, proposta por Mendes *et al.* [18], para o ETPM não preemptivo.

A classe de problema abordada neste trabalho 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.

4.2 ESCALONAMENTO EM TEMPO REAL

Conceitos e técnicas de escalonamento em tempo real formam o ponto central na previsibilidade do comportamento de sistemas de tempo real em sistemas onde as noções de tempo e de concorrência são tratadas explicitamente. Nos últimos anos, uma

quantidade significativa de novos algoritmos e de abordagens foi introduzida na literatura tratando de escalonamento de tempo real. Infelizmente muitos desses trabalhos definem técnicas restritas e conseqüentemente de uso limitado em aplicações reais.

Segundo Farines, Fraga e Oliveira [3], as aplicações de tempo real são caracterizadas por restrições temporais que devem ser respeitadas para que se tenha o comportamento temporal desejado ou necessário. Todas as tarefas de tempo real tipicamente estão sujeitas a prazos: os seus *deadlines*. A princípio, uma tarefa deve ser concluída antes de seu *deadline*. As conseqüências de uma tarefa ser concluída após o seu *deadline* definem dois tipos de tarefas de tempo real:

• Tarefas Críticas (*hard*): uma tarefa é dita crítica quando ao ser completada depois de seu *deadline* pode causar falhas catastróficas no sistema de tempo real e em seu ambiente. Essas falhas podem resultar em danos irreversíveis em equipamentos ou ainda, em perda de vidas humanas.

• Tarefas Suaves ou Não Críticas (*soft*): Essas tarefas quando se completam depois de seus *deadlines* no máximo implicam numa diminuição de desempenho do sistema. As falhas temporais nesse caso são identificadas como benigna onde a conseqüência do desvio do comportamento normal não representa um custo muito significativo.

A característica temporal de tarefas em sistemas de tempo real está baseada na regularidade de suas ativações. Seus modelos comportam dois tipos de tarefas baseados em suas freqüências de ativações: periódicas e aperiódicas, onde, na primeira, as tarefas têm períodos pré-determinados para serem executadas e, na segunda, não há um tempo específico de início de execução da tarefa e, este tempo, pode ser definido a qualquer momento no sistema.

O Escalonamento de tempo real considera, como restrições temporais das tarefas, o tempo de computação de uma tarefa, tempo de início e término da tarefa, tempo de chegada da tarefa e tempo de liberação da tarefa (*release time*), sendo que, este último pode ou não coincidir com o tempo de chegada da tarefa. Dessa forma, calcula-se o *Release Jitter*, que corresponde ao tempo entre o instante de chegada da tarefa e o release time da mesma.

O tipo de escalonamento tratado neste trabalho é o de tempo real com tarefas aperiódicas, onde são consideradas, como restrições temporais, o tempo de computação de uma tarefa e o tempo de liberação da tarefa (*release time*).

4.3 ESCALONAMENTO POR PRIORIDADES

Há uma grande variedade de suportes, na forma de produtos, com escalonamentos baseados em mecanismos de prioridade.

Este tipo de escalonamento tem sido um dos mais usados nos escalonamentos em tempo real e tem suas prioridades definidas segundo critérios de escalonamentos próprios para cada problema. Essas prioridades podem ser definidas como fixas (estáticas) ou variáveis (dinâmicas), de acordo com as restrições do problema a ser estudado. Dessa forma, o escalonamento é realizado *on-line* e seus parâmetros podem ser dinâmicos.

O problema de escalonamento abordado neste trabalho é baseado em mecanismos de prioridades estáticas (fixas).

4.4 ESCALONAMENTO USADO NO TRABALHO

O problema de escalonamento abordado neste trabalho é baseado em mecanismos de prioridades estáticas (fixas), em tempo real, considerando que as tarefas são aperiódicas, tendo-se, como restrições temporais, o tempo de computação de uma tarefa e o tempo de liberação da mesma (*release time*).

A classe de problema abordada neste trabalho é definida como sendo um conjunto de *n* tarefas que deve ser organizado em *m* máquinas não idênticas. O tempo de *setup* é independente, 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.

5 – MÉTODOS CONSTRUTIVOS E DE BUSCA LOCAL GERADOS PARA O PROBLEMA

Neste capítulo serão descritos todos os algoritmos de construção e de busca local criados para o problema de escalonamento abordado.

5.1. MÉTODOS CONSTRUTIVOS

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

Vale ressaltar que sempre será gerada uma solução viável (possível), pois em todos os algoritmos são considerados os release time das embarcações.

5.1.1 Escalonamento de tarefas por prioridades (ETP)

O algoritmo ETP considera que, quanto menor o valor numérico da prioridade de 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.

O algoritmo apresentado na Figura 3 recebe como entrada, o número de embarcações (*N_emb*) existentes na fila de espera do porto, a lista de prioridade (prior) e

o release time (rt) de cada embarcação e apresenta como saída a solução (*Sol*) com a fila de espera escalonada pelo método aplicado. Quando se tem mais de uma tarefa com o mesmo grau de prioridade classifica-se, em ordem crescente, pelo *release time (*tempo estimado para início de execução da tarefa) da mesma, conforme é descrito na linha 1. No laço das linhas 5-11, é gerado a solução inicial incluindo-se cada tarefa em uma das máquinas que podem atendê-la. Quando uma tarefa puder ser executada por mais de uma máquina, escolhe-se aleatoriamente a máquina a executá-la.

Procedimento ETP (*N_emb, prior ,rt*)
Entrada
N_emb - número de embarcações na fila de espera;
prior – Prioridade das embarcações.
rt – Release time das embarcações.

Saída
Sol – a ordem de atendimento das embarcações em cada píer gerada na solução construtiva.
Início

01. Classifique as embarcações em ordem crescente de prior e rt , e armazene em *LEmb*.
02. **Para** *i* **de** 1 **até** *N_emb* **faça**
03. **Início**
04. *qb* ⟵ quantidade de berços que podem atender o barco *LEmb*(i);
05. **se** *qb* ⟵ 1 **então**
06. inclua o barco *LEmb*(i) em *Sol* no berço que pode atendê-lo, obedecendo o rt;
07. **senão**
08. **Início**
09. r ⟵ escolhe aleatoriamente um berço;
10. inclua barco *LEmb*(*i*) em *Sol*, no berço *r* ;
11. **Fim-senão**;
12. **Fim-para;**
13. Calcula atraso em *Sol* usando a fórmula $\sum_{n\_emb}^{i=1} pp_i \times te_i = \sum_{n\_emb}^{i=1} pp \times (at_i - rt_i)$;

Fim_ETP;

Figura 3 – Algoritmo ETP desenvolvido.

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

Para essa solução construtiva foi considerada, primeiramente, a quantidade de máquinas que podem executar determinada tarefa, tomando-se como prioridade as tarefas que têm menos opções de máquinas para executá-las.

O algoritmo da Figura 4 recebe como parâmetros de entrada o número de embarcações na fila de espera (*N_emb*), a lista de prioridade (prior), o release time (rt) de cada embarcação, a quantidade de tarefas (*n*) as quais serão feitas a aleatoriedade para a inclusão de cada barco e a quantidade de berços que atendem cada barco (*qb*). O método apresentará, como saída, a solução (*Sol*), com a lista de soluções construídas pelo método.

.Na linha 1, as tarefas são classificadas, em ordem crescente de quantidade de máquinas que podem atendê-las. Assim, as tarefas que só podem ser atendidas por uma máquina serão executadas primeiro, depois as que podem ser atendidas por duas máquinas e assim por diante. Quando mais de uma tarefa possui a mesma quantidade de máquinas para atendê-la, a ordem de classificação será pelo grau de prioridade das tarefas (considerando-se a ordem crescente das prioridades, conforme descrito anteriormente).

As tarefas são, então, inseridas nas máquinas escolhendo-se aleatoriamente entre as *n* primeiras tarefas, como mostrado na linha 4.

Ao se inserir uma tarefa na máquina, deve-se escolher a melhor posição considerando todas as posições disponíveis em todas as máquinas que podem atender a tarefa, a partir do *release time* da mesma, de forma que o atraso total final na máquina seja o menor possível, como é feito no laço das linhas 6-15, depois, inserida a tarefa na linha 16.

O vetor de tarefas (no algoritmo, *Lemb*) deve ser reconstruído a cada tarefa que é retirada da fila de espera de forma que, a cada iteração, a quantidade total de tarefas a serem executadas seja diminuído de 1 unidade (linha 17). O atraso total gerado é calculado na linha 19.

Procedimento ETQM1 (*N_emb, prior, rt, n,qb*)
Entrada
N_emb - número de embarcações na fila de espera.
n – quantidade de tarefas a ser feita a aleatoriedade.
qb – quantidade de berços que podem atender cada barco.
prior – Prioridade das embarcações.
rt – Release time das embarcações.

Saída
Sol – a ordem de atendimento das embarcações em cada píer gerada na solução construtiva.
Início
01. Classifique a lista de embarcações em ordem crescente de qb e prior e coloque em *LEmb*;
02. **Para** *i* **de** 1 **até** *N_emb* **faça**
03. **Início**
04. Selecione aleatoriamente um barco *b* entre os *n* primeiros barcos;
05. *melhor_atraso* ⟵ ∞;
06. **Para** cada berco que pode atendê-lo
07. **Início**
08. *pos* ⟵ melhor posição livre no berço respeitando o rt;
09. *atraso* ⟵ atraso no berço se *b* for incluído na posição *pos*;
10. **se** *atraso* < *melhor_atraso* **então**
11. **Inicio**
12. *melhor_berco* ⟵ *berco*;
13. *melhor_pos* ⟵ *pos*;
14. **Fim-se;**
15. **Fim-para;**
16. inclua barco *LEmb*(*i*) no berço *melhor_berco*, na posição *melhor_pos* em *Sol*;
17. Retira *b* de *LEmb*;
18. **Fim-para;**
19. Calcula atraso total em *Sol usando a fórmula* $\sum_{n\_emb}^{i=1} pp_i \times te_i = \sum_{n\_emb}^{i=1} pp \times (at_i - rt_i)$;
Fim_ETQM1;

Figura 4 – Algoritmo ETQM1 desenvolvido.

5.1.3 Escalonamento de tarefas por quantidade de máquinas que podem executá-las, onde, se a posição inicial esteja ocupada, considere a tarefa de melhor prioridade (ETQM2)

Procedimento ETQM2 (*N_emb, prior, rt, n, qb*)
Entrada
N_emb - número de embarcações na fila de espera.
n – quantidade de tarefas a ser feita a aleatoriedade.
qb – quantidade de berços que pode atender cada barco.
prior – Prioridade das embarcações.
rt – Release time das embarcações.

Saída
Sol - a ordem de atendimento das embarcações em cada píer gerada na solução construtiva.
Início
01. Classifique a lista de embarcações em ordem crescente de qb e rt;
02. **Para** *i* **de** 1 **até** *N_emb* **faça**
03. **Início**
04 Selecione aleatoriamente um barco *b* entre os *n* primeiros barcos de *LEmb*;
05. m*elhor_atraso* ⟵ ∞;
06. **Para** cada berço que pode atendê-lo
07. **Início**
08. **se** posição do rt de b estiver livre ou
prioriopridade do barco *b* < Prioridade do barco que está no berço **então**
09. **inicio**
10. *pos* ⟵ rt *b*;
11. *atraso* ⟵ 0;
12. **Fim-se**;
13. **senão**
14. **inicio**
15. *pos* ⟵ primeira posição a seguir do barco que está no píer;
16. *atraso* = atraso no berço se barco for incluído no berço na posição *Pos*;
17. **Fim-senão**;
18. **se** atraso < melhor_atraso **então**
19. **Inicio**
20. *melhor_berco* ⟵ *berco*;
21. *melhor_pos* ⟵ *pos*;
22. **Fim-se**;
23. inclui barco *b* no berço *melhor_berco*, na posição *melhor_pos* em *Sol*;
24. **Fim-para**;
25. Calcula atraso total em *Sol usando a fórmula* $\sum_{n\_emb}^{i=1} pp_i \times te_i = \sum_{n\_emb}^{i=1} pp \times (at_i - rt_i)$;

Fim_ETQM2;

Figura 5 – Algoritmo ETQM2 desenvolvido.

A Figura 5 mostra o algoritmo para essa estratégia de escalonamento, onde foram considerados, basicamente, os mesmos critérios do escalonamento anterior. A diferença está no fato de que, se a posição do *release time* da tarefa já estiver ocupada

por outra tarefa na máquina, considera-se a que tiver a melhor prioridade. Dessa forma, como é mostrado na linha 8, se a tarefa a entrar na maquina, tiver prioridade melhor que a que já está na máquina, esta empurrará a tarefa da máquina para frente e entrará antes dela. Caso contrário a tarefa a ser inserida será colocada logo após a tarefa que está na máquina, como é feito no laço das linhas 14-17.

Analogamente à heurística anterior, o laço das linhas 6-24 verifica que, 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. Na linha 25 é calculado o atraso total gerado.

5.2 ALGORITMOS DE BUSCA LOCAL

Aqui serão apresentados os algoritmos de busca local implementados para o problema.

5.2.1 Busca local intercambio(BLI)

Na Figura 6 é mostrado o pseudocódigo de um movimento desta Busca Local, onde a rotina recebe como parâmetro de entrada o número de embarcações(*N_emb*) e a lista de embarcações a ser refinada (*LEmb*) e, como saída, a própria lista *LEmb*, porém refinada com a melhor troca encontrada na busca local. O laço entre as linhas 3-23 realiza a busca local na solução inicial, considerando todas as trocas possíveis entre 2 embarcações de berços diferentes. Para que a troca seja analisada verifica-se se ambas as embarcações podem trocar de berços, ou seja, para que a troca da embarcação *a,* que está no píer *X,* pela embarcação *b,* que está no píer *Y* possa ser analisada, é necessário que o barco *a* possa ser alocado no píer *Y* e o barco *b* possa ser alocado no píer *X,* como é verificado na linha 10. Se a condição for satisfeita, a troca dos barcos é analisada calculando-se o novo atraso gerado nos berços *X* e *Y.* Se os atrasos obtidos forem melhores que os atuais nos berços, aceita-se a troca como melhor (linhas 14 a 19) e a busca local continua até que todas as trocas possíveis sejam analisadas. O procedimento BLI realiza buscas enquanto houver melhora na solução corrente ou até que todas as embarcações sejam analisadas. A melhor troca obtida só será considerada

para o resultado final se o melhor atraso for negativo, ou seja, houve diminuição em relação ao atraso atual no berço, caso contrário, o resultado obtido é descartado.

```
Procedimento BLI ( N_emb, LEmb )
Entrada
    N_emb - número de embarcações na fila de espera.
    LEmb – a lista de embarcações criada por um método construtivo.

Início
01.  Método construtivo (LEmb);
02. Melhor_atraso ⟵ -1;
03.  Enquanto Melhor_atraso  < 0 faça
04.  Início
05.       Para i=1 até N_emb faça
06.          Para j=i+1 até N_emb faça
07.          Início
08.            bercoi ⟵ berço do barco i;
09.            bercoj ⟵ berço do barco j;
10.            Se (bercoi) <> (bercoj)
               e (barco i pode ir para o berçoj)
               e (barco j pode ir para o berçoi) então
11.             Início
12.               Atrasoi ⟵ atraso no bercoi se barco i trocar com o barco j;
13.               Atrasoj ⟵ atraso no bercoj se barco j trocar com o barco i;
14.               Se (Atrasoi + Atrasoj ) < Melhor_atraso então
15.                Início
16.                  Melhori ← i;
17.                  Melhorj ← j;
18.                  Melhor_atraso  ← Atrasoi + Atrasoj;
19.                Fim-se;
20.             Fim-se;
21.          Fim-para;
22.          Se melhor_atraso < 0 então Troca melhori com melhorj em LEmb;
23.  Fim-enquanto;
24.  Retorne LEmb;
Fim_BLI;
```

Figura 6 – Algoritmo BLI desenvolvido

5.2.2 Busca local troca (BLT)

Para a BLT, o método é semelhante ao da BLI, diferindo, apenas, no fato de que serão consideradas somente as trocas entre embarcações de mesmo berço conforme é mostrado na linha 10, ou seja, haverá, um intercâmbio dos barcos de um mesmo berço. Dessa forma, o algoritmo construído para a BLT é mostrado na Figura 7. O procedimento BLT realiza a troca enquanto houver melhora na solução. A melhor troca obtida só será

considerada para o resultado final se o melhor atraso for negativo, ou seja, houve diminuição em relação ao atraso atual no berço, caso contrário, o resultado obtido é descartado.

```
Procedimento BLT ( N_emb, LEmb )
Entrada
    N_emb - número de embarcações na fila de espera.
    LEmb – a lista de embarcações criada por um método construtivo.

Início
01.  Método construtivo (LEmb);
02. Melhor_atraso ⟵ -1;
03.  Enquanto Melhor_atraso  < 0 faça
04.   Início
05.      Para i=1 até N_emb faça
06.       Para j=i+1 até N_emb faça
07.       Início
08.          bercoi ⟵ berço do barco i;
09.          bercoj ⟵ berço do barco j;
10.           Se (bercoi) =(bercoj)  então
11.           Início
12.              Atraso ⟵ atraso no bercoi se barco i trocar com o barco j;
13.              Se (Atraso ) < Melhor_atraso então
14.              Início
15.                 Melhori ⟵ i;
16.                 Melhorj ⟵ j;
17.                 Melhor_atraso  ⟵ Atraso;
18.              Fim-se;
19.           Fim-se;
20.       Fim-para;
21.       Se melhor_atraso < 0 então Troca melhori com melhorj;
22.   Fim-enquanto;
23.        Retorne LEmb;
Fim_BLT;
```

Figura 7 – Algoritmo BLT desenvolvido.

5.2.3 Busca local realocação (BLR)

Nesta busca foi considerada a realocação de, uma embarcação para um outro berço. O algoritmo d BLR é apresentado na Figura 8, onde recebe como parâmetro o número de embarcações na fila de espera (*N_emb*), a lista de embarcações a ser refinada (*LEmb*) e o número de berços existentes no porto (N_berco) e gera, como saída a lista (*LEmb*) refinada com a melhor realocação encontrada. Neste algoritmo não há troca entre duas embarcações como foi considerada nos dois algoritmos de busca local implementados anteriormente. No laço das linhas 3-20 é realizada a busca em todas as embarcações em todos os berços e nas linhas 13, 14 e 15 são armazenados os

parâmetros com a melhor solução encontrada. Na linha 19 a realocação é realizada com a melhor solução encontrada, somente se o atraso obtido for negativo, ou seja, houve diminuição em relação ao atraso atual no berço, caso contrário, o resultado obtido é descartado. O procedimento BLR realiza a busca enquanto houver melhora na solução corrente.

```
Procedimento BLR ( N_emb, LEmb,N_berco )
Entrada
    N_emb - número de embarcações na fila de espera;
    LEmb – a lista de embarcações criada por 1 método construtivo;
    N_berco – número de berços existentes no porto
 Início
01. Método construtivo (LEmb);
02. Melhor_atraso ← -1;
03. Enquanto Melhor_atraso  < 0 faça
04. Início
05.     Para i=1 até N_emb faça
06.     Início
07.       bercoi ← berço do barco i;
08.       Para j=1 até N_berco faça
09.       Início
10.             Atraso ← atraso se barco i for para berço j;
11.             Se (Atraso ) < Melhor_atraso então
12.             Início
13.                Melhori ← i;
14.                Melhorberco ← j;
15.                Melhor_atraso  ← Atraso;
16.             Fim-se;
17.       Fim-se;
18.     Fim-para;
19.     Se Melhor_atraso < 0 então Troca melhori para melhorberco;
20. Fim-enquanto;
21. Retorne LEmb;
Fim_BLR;
```

Figura 8 – Algoritmo BLR desenvolvido.

6 – A METAHEURÍSTICA GRASP

O *Greedy Randomized Adaptive Search Procedure* (GRASP) é uma metaheurísticas de múltiplas partidas proposta por Feo e Resende [26], onde a cada iteração é gerada uma solução gulosa-aleatória – fase construtiva, que em seguida é melhorada por uma heurística de busca local – fase de melhoria. Na fase construtiva, é construída iterativamente uma solução viável, inserindo na solução parcial um elemento de cada vez segundo um critério e aleatório. A cada iteração da fase construtiva são avaliados apenas elementos que podem ser adicionados à solução sem violar as restrições de viabilidade, gerando assim sempre uma solução factível. Esses elementos são chamados de elementos candidatos. A escolha do próximo elemento a ser adicionado a solução é determinada ordenando-se todos os elementos candidatos em uma lista de candidatos C, de acordo com uma função gulosa. Essa função mede o benefício associado à seleção de cada elemento. A heurística é adaptativa porque os benefícios associados a cada elemento são atualizados a cada iteração da fase construtiva, para incorporar as mudanças causadas pela escolha do último elemento. A componente probabilística é caracterizada pela escolha aleatória de um dos melhores candidatos da lista C, que não é necessariamente o melhor. A lista de melhores candidatos é denominada de lista restrita de candidatos (LRC).

Sumarizando, a estratégia da metaheurísticas GRASP consiste em usar diferentes soluções iniciais como pontos de partida para ser melhorada por uma busca local. Uma solução *x* é dita como pertencente ao vale de ótimo local quando, a partir de uma busca local iniciada em *x*, é possível atingir este ótimo local. Caso uma das soluções iniciais esteja no vale de um ótimo global, a busca local irá encontrar este ótimo global. Caso contrário, a solução do algoritmo será um ótimo local. O uso de diversos pontos de partidas aleatórios permite eventualmente ao algoritmo encontrar um ponto dentro do vale de um ótimo global, porém as soluções de partida aleatórias

geralmente são de baixa qualidade e acaba sendo necessário um grande número de movimentos para que se ache um ótimo local ou global. Por um outro lado, as soluções gulosas apresentam boas soluções como ponto de partida, no entanto os mesmos sempre param em um ótimo local quando aplicado um procedimento de busca local. Para utilizar a diversidade das soluções aleatórias e qualidade inicial das soluções produzidas pelos algoritmos gulosos, o GRASP utiliza um fator alfa de aleatoriedade, onde quanto maior esse fator maior é a aleatoriedade da solução de partida.

Assim, foram desenvolvidas neste trabalho heurísticas GRASP para resolver o problema de escalonamento de tarefas em máquinas paralelas com o objetivo de definir qual destas heurísticas se adapta melhor ao problema em questão.

A Figura 9 apresenta o pseudocódigo da heurística GRASP desenvolvida 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 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*.

```
Procedimento GRASP ( N_iter )
Entrada
    N_iter - número de iterações GRASP;
Saída
    Sol – melhor ordem de atendimento das embarcações em cada píer encontrada.
Início
01.     Para i de 1 até N_iter faça
02.     Início
03.         s ← Construção (α);
04.         s' ← BuscaLocal (s);
05.       Se s' for a melhor solução até o momento então
06.           Sol ← s';
07.     Fim_para
08.     Retorne Sol;
Fim_GRASP
```

Figura 9 –. Algoritmo GRASP desenvolvido.

6.1 ALGORITMOS GRASP

Os algoritmos GRASP foram montados fazendo-se combinações entre os métodos construtivos e as buscas locais propostos. Dessa forma, os métodos GRASP`s propostos foram o seguintes:

GRASPEPBT – ETP + BLT
GRASPE1BT – ETQM1 + BLT
GRASPE2BT – ETQM2 + BLT
GRASPEPBI – ETP + BLI
GRASPE1BI – ETQM1 + BLI
GRASPE2BI – ETQM2 + BLI
GRASPEPBR – ETP + BLR
GRASPE1BR – ETQM1 + BLR
GRASPE2BR – ETQM2 + BLR

6.1.1 Resultados obtidos com as heurísticas GRASP

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. A média geral dos resultados alcançados é apresentada na Tabela 2.

A máquina usada para realização dos experimentos foi um notebook Compaq Presario R3000 com processador Atlhon 3200 com 512Mb de memória. Os algoritmos GRASP aqui propostos foram implementados em C utilizando a versão 6.0 do compilador Microsoft Visual C++.

Pelos resultados obtidos, observa-se que o algoritmo GRASPE1BT e GRASPE1BI foram os que obtiveram os melhores resultados na maioria dos problemas apresentados. No entanto, o tempo computacional obtido não foram os melhores. Baseado nisso, foi realizado um novo experimento, nos mesmos moldes, porém, fixando-se o tempo de execução, tomando-se como base o maior tempo obtido em todos os métodos GRASP do primeiro experimento para cada problema teste. A nova média geral dos resultados alcançados é apresentada na Tabela 3.

Os tempos computacionais, para este novo experimento, foram fixados em:

Ins5-48 = 76; Ins5-72 = 116;
Ins5-96 = 218; Ins5-120 = 404;
Ins6-48 = 123; Ins6-72 = 194;
Ins6-96 = 415; Ins6-120 = 649;
Ins7-48 = 163; Ins7-72 = 304;
Ins7-96 = 574; Ins7-120 = 991;
Ins8-48 = 205; Ins8-72 = 468;
Ins8-96 = 823; Ins8-120 = 1294;

A partir destes novos resultados obtidos, conclui-se que as heurísticas GRASPE1BT e GRASPE1BI ainda continuam sendo as que apresentaram os melhores resultados. Como ambas utilizam, como método construtivo, a ETQM1, deduz-se que este é o melhor método construtivo aplicado às heurísticas GRASP implementadas neste trabalho.

	GRASPEPBT		GRASPE1BT		GRASPE2BT		GRASPEPBI		GRASPE1BI		GRASPE2BI		GRASPEPBR		GRASPE1BR		GRASPE2BR	
	Atraso	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**
Ins5-48	3225	10,78	340	75,68	346	21,86	4536	2	338,6	69,42	338,2	12,88	2411	4,32	359,4	64,86	363	15,16
Ins5-72	2312,4	21,54	605,8	116,04	633,4	41,5	3118,6	4,48	568,6	100,12	579,4	20,92	2646,2	7	680,6	92,66	669,4	21,82
Ins5-96	1925	89,6	1059	217,38	1128,4	122,74	2651,4	8,62	1050,2	146,42	1114,4	34,7	2055,4	14,18	1110,4	138,7	1136	39,84
Ins5-120	2697,6	172,92	1565	403,9	1573,6	177,26	3507,8	17,02	1525,8	296,5	1651,8	49,9	3067,8	18,88	1598,8	265,38	1717,2	46,96
Ins6-48	1520	19,72	427,8	122,86	521,8	42,4	2623	2,94	431	108,68	466	22,52	1520	6,3	440,6	101,02	523,4	25,12
Ins6-72	2952,8	53,86	1160,2	193,18	1192	76,04	4438,2	7,54	1161	159,6	1176	34,58	2951,4	10,78	1170,2	146,24	1212,4	36,18
Ins6-96	3789	143,8	1688,2	414,84	1864	168,6	4179,8	16,42	1665,4	319,8	1857,2	56,2	3770,2	19,1	1707	288,1	1923	57,86
Ins6-120	3640,8	224,52	1000,4	648,36	1304,6	258,98	4067	28,2	1018,4	502,8	1291	77,48	3638,6	23,46	1015,6	449,06	1346,8	70,38
Ins7-48	1153	36,32	561	162,58	645	53,38	1185,6	4,28	570,2	145,66	633,8	32,2	1195,4	10,76	573,4	135,8	675	36,76
Ins7-72	4226	105,56	1554,6	304,12	1258,2	121,22	4910,8	10,86	1218,8	249,02	1229	56,52	3250,8	16,8	1602,8	230,72	1331,8	58,08
Ins7-96	4183,4	291,54	2102	573,26	2299,4	310,74	6566	22,86	2150,2	379,22	2314,8	93,32	5209	31,5	2261,2	347,5	2398,2	76,28
Ins7-120	6434	436,66	2159,8	990,42	2194,8	435,4	8131,2	43,36	2199,8	732,96	2087,2	133,38	6497,4	37,8	2327,4	640,2	2198,8	112,42
Ins8-48	1232,6	37,62	786	204,9	932,8	64,2	1329,4	5,76	790,2	190,44	972	40,22	1217,8	12,14	804,6	175	1017	43,74
Ins8-72	4072,2	171,56	1830,2	467,86	2293,4	187,42	4015,6	17,02	1817,6	386,06	2321,8	81,08	3770	25,72	1873,8	353,26	2399,6	68,94
Ins8-96	4248,8	312,8	1725,4	822,98	2002	358,44	4729,4	36,44	1704,4	638,82	2045	123,46	4236,2	37,58	1759	567,52	2138,4	118,2
Ins8-120	5323,4	504,76	1959,6	1293,28	2205,6	583,8	6364	60,9	1992,6	933,14	2366	170,5	5926,2	51,8	2083,6	824,18	2432,6	154

Tabela 2 – Resultados GRASP com a média geral.

	GRASPEPBT	GRASPE1BT	GRASPE2BT	GRASPEPBI	GRASPE1BI	GRASPE2BI	GRASPEPBR	GRASPE1BR	GRASPE2BR
	Atraso	**Atraso**	**Atraso**	**Atraso**	**Atraso**	**Atraso**	**Atraso**	**Atraso**	**Atraso**
Ins5-48	3225	340	337,4	4536	338,6	327,4	2411	359,4	350,8
Ins5-72	1930,4	605,8	630,8	2995,6	568,6	577	2490,2	646	664,4
Ins5-96	1837,6	1059	1119,4	2343,2	1050,2	1062,6	1882,6	1103,2	1093,4
Ins5-120	2573	1565	1565,8	3168	1525,8	1570	2816,6	1598,8	1634
Ins6-48	1520	427,8	502,4	2623	431	458	1520	440,6	496,8
Ins6-72	2941	1160,2	1172,6	4431	1159	1156,8	2941	1169	1187,6
Ins6-96	3749,8	1696,6	1831	4050,2	1665,4	1811	3650	1705,4	1877,6
Ins6-120	3470,4	1000,4	1301	3601,8	1011,6	1255,2	3068,6	1004,6	1311,2
Ins7-48	1088,8	561	630,4	1075,2	570,2	619,2	1088,8	573,4	651
Ins7-72	4081,2	1554,6	1252	4666,4	1218,8	1209,8	3084,4	1602,8	1322,4
Ins7-96	4183,4	2102	2276	5844,2	2103	2269,2	4898	2256,8	2345,6
Ins7-120	6349,6	2159,8	2160	7446,8	2184,4	2053,4	5894,6	2289	2155,4
Ins8-48	1137,4	786	916,4	1146	787,2	939	1083	804,6	999
Ins8-72	3987,4	1830,2	2270,8	3683	1817,6	2265,2	3569	1873,8	2339,2
Ins8-96	4192,2	1725,4	1956,6	4455	1700,6	1969,2	3964	1725	2048,6
Ins8-120	5244	1959,6	2192,6	6041	1987,4	2292,2	5733	2077,2	2347,4

Tabela 3 – Resultados GRASP considerando tempo de execução

7 – ESTRATÉGIAS GRASP + VND PROPOSTAS

O método VND (*Varible Neighborhood Descent*) é estratégia de refinamento de soluções, que se difere dos métodos de busca local tradicionais por trabalhar com mais de uma estrutura de vizinhança.

Sejam $N_1, N_2, ..., N_p$ estruturas de vizinhança tal que $N_k(x)$ é o conjunto de soluções na *k*-ésima vizinhança da solução *x*. Normalmente é assumido que a vizinhança N_{k+1} possui mais elementos do que a vizinhança N_k. Abaixo é apresentado o pseudocódigo do algoritmo VND padrão. O laço nas linhas 3-13 é repetido até que todas as vizinhanças sejam analisadas sem sucesso. Na linha 5 uma busca local na vizinhança N_k *é* aplicada a *x*, gerando a solução *x'*. Se *x'* for melhor do que *x* então a solução *x' é* atribuída a *x* e o processo é reiniciado com a vizinhança N_1, como é mostrado nas linhas 8 e 9. Caso contrário, o processo continua com a vizinhança N_{k+1}.

```
Procedimento VND (x0, N1,...Np )
Entrada
    x0 -uma solução viável para o problema.
    N1,...Np - as vizinhanças.
Saída
    x – melhor solução encontrada.
Início
01. x ⟵ x0;
02. k ⟵ 1;
03. Enquanto  k ≥ p faça
04.     Início
05.         Aplique uma busca local em x usando a vizinhança Nk. Seja x' o ótimo local;
06.         Se f(x') < f(x) então
07.         Inicio
08.             x ⟵ x' ;
09.              k ⟵ 1;
10.         Fim-se;
11.         Senão k ⟵ k+1;
12.          Sol ⟵ s';
13. Fim-enquanto;
14.     Retorne x;
Fim
```

Figura 10 – Algoritmo VND padrão.

Conforme concluído no capítulo anterior o método ETQM1, aplicado na fase de construção das heurísticas GRASP, foi o que apresentou os melhores resultados. Assim, as novas heurísticas implementadas foram construídas usando-se, então, o método construtivo ETQM1 e as estratégias VND (utilizando-se as buscas locais até então implementadas), obtendo-se, assim, novas heurísticas, como será descrito a seguir.

7.1 HEURÍSTICAS USANDO O ALGORITMO CONSTRUTIVO ETQM1 + VND (HVND)

Para estas heurísticas foi usado o método construtivo ETQM1 (visto no Capítulo 4, Seção 4.1.2) juntamente com as 3 buscas locais implementadas neste trabalho (BLT, BLI e BLR). Para se obter o melhor resultado, foram implementados seis algoritmos VND's combinando-se as 3 buscais locais da seguinte forma:

HVNDITR:

– Construção: ETQM1

– N_1: BLI

– N_2: BLT
– N_3: BLR

HVNDRIT:
– Construção: ETQM1
– N_1: BLR
– N_2: BLI
– N_3: BLT

HVNDIRT:
– Construção: ETQM1
– N_1: BLI
– N_2: BLR
– N_3: BLT

HVNDTRI
– Construção: ETQM1
– N_1: BLT
– N_2: BLR
– N_3: BLI

HVNDRTI:
– Construção: ETQM1
– N_1: BLR
– N_2: BLT
– N_3: BLI

HVNDTIR
– Construção: ETQM1
– N_1: BLT
– N_2: BLI
– N_3: BLR

7.2 RESULTADOS OBTIDOS COM AS HEURÍSTICAS USANDO O MÉTODO GRASP + VND

Na Tabela 4 são apresentados os resultados obtidos pelos algoritmos GRASP+VND propostos. No experimento realizado, cada algoritmo GRASP + VND proposto foi executado durante *N_iter*=1000 iterações para cada Problema teste descrito na Tabela 1 (Capítulo 5, Seção 5.1.1). Este processo foi repetido cinco vezes, variando a semente de geração de números aleatórios. A média geral dos resultados alcançados é apresentada na Tabela 4.

	VNDITR		VNDRIT		VNDIRT		VNDTRI		VNDRTI		VNDTIR	
	Atraso	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**	**Atraso**	**Tempo**
Ins5-48	327,8	80,06	327,8	76,88	327,8	78,98	327,8	76,34	327,8	77,2	327,8	76,22
Ins5-72	568,6	125,46	568,6	120,72	568,6	123,72	568,6	117,94	568,6	120,28	568,6	118,08
Ins5-96	992,2	234,06	992,2	224,16	992,2	231,3	997,4	224,36	997,4	224	997,4	224,88
Ins5-120	1423,0	420,32	1423,0	401,66	1423,0	415,22	1433,2	404	1433,2	400,98	1433,2	405,3
Ins6-48	421,0	128,58	421,0	122,7	421,0	127,02	421,0	123,4	421,0	207,92	421,0	124,16
Ins6-72	1147,8	204,98	1147,8	196,08	1147,8	202,42	1147,8	196,62	1147,8	196,38	1147,8	197,26
Ins6-96	1623,2	432,52	1623,2	413,2	1623,2	427,08	1622,4	414,7	1622,4	413,22	1622,4	416,64
Ins6-120	958,6	662,88	958,6	632,04	958,6	653,78	958,6	632,8	958,6	631,84	958,6	635,52
Ins7-48	560,8	172,76	560,8	164,52	560,8	170,1	560,8	164,08	560,8	164,28	560,8	164,78
Ins7-72	1136,4	318,38	1136,4	302,92	1136,4	313,24	1115,6	301,56	1115,6	302,78	1115,6	302,84
Ins7-96	1968,0	615,42	1968,0	584,28	1968,0	604,02	1945,0	582,58	1945,0	582,14	1945,0	591,16
Ins7-120	1987,2	1047,6	1987,2	993,22	1987,2	1026,98	1899,4	990,78	1899,4	990,6	1899,4	1025,08
Ins8-48	767,8	216,12	767,8	204,96	767,8	211,96	767,8	204,82	767,8	204,82	767,8	213,18
Ins8-72	1770,0	500,9	1770,0	475,06	1770,0	491,22	1764,8	475,62	1764,8	474,6	1764,8	487,38
Ins8-96	1657,8	872,8	1657,8	827,62	1657,8	853,96	1636,2	832,06	1636,2	827,18	1636,2	833,6
Ins8-120	1822,8	1383,24	1822,8	1304,94	1822,8	1352,22	1768,2	1292	1768,2	1300,6	1768,2	1305,86

Tabela 4 – Média dos resultados dos algoritmos VND's com 3 buscas locais.

Pelos resultados obtidos, observa-se que os algoritmos VNDTRI, VNDRTI e VNDTIR foram os que obtiveram os melhores resultados na maioria dos problemas apresentados. Seguindo os mesmos testes dos algoritmos GRASP, foi realizado um novo experimento, fixando-se o tempo de execução, tomando-se como base o maior tempo obtido em todos os métodos GRASP + VND no primeiro caso para cada problema teste. A média geral dos resultados alcançados variando-se as sementes da mesma forma que no caso anterior e executando-se por tempo de execução é apresentada na Tabela 5, juntamente com os algoritmos GRASPE1BT e GRASPE1BI, agora executados com os mesmos tempos usados nos métodos GRASP + VND.

	VNDITR	VNDRIT	VNDIRT	VNDTRI	VNDRTI	VNDTIR	GRASPE1BT	GRASPE1BI
	Atraso	**Atraso**	**Atraso**	**Atraso**	**Atraso**	**Atraso**	**Atraso**	**Atraso**
Ins5-48	327,8	327,8	327,8	327,8	327,8	327,8	340	338,6
Ins5-72	568,6	568,6	568,6	568,6	568,6	568,6	597,4	568,6
Ins5-96	992,2	992,2	992,2	997,4	997,4	997,4	1059	1050,2
Ins5-120	1423	1423	1423	1433,2	1433,2	1433,2	1565	1523,2
Ins6-48	420,8	420,8	420,8	420,8	420,8	420,8	426,4	428,8
Ins6-72	1147,8	1147,8	1147,8	1147,8	1147,8	1147,8	1155	1159
Ins6-96	1623,2	1623,2	1623,2	1622,4	1622,4	1622,4	1688,2	1663,8
Ins6-120	958,6	958,6	958,6	958,6	958,6	958,6	1000,4	1011,6
Ins7-48	560,8	560,8	560,8	560,8	560,8	560,8	561	570,2
Ins7-72	1136,4	1136,4	1136,4	1115,6	1115,6	1115,6	1554,6	1214,4
Ins7-96	1968	1968	1968	1945	1945	1945	2102	2103
Ins7-120	1987,2	1987,2	1987,2	1899,4	1899,4	1899,4	2143,2	2161,2
Ins8-48	767,8	767,8	767,8	767,8	767,8	767,8	786	787,2
Ins8-72	1770	1770	1770	1764,8	1764,8	1764,8	1824	1817,6
Ins8-96	1657,8	1657,8	1657,8	1636,2	1636,2	1636,2	1725,4	1693,2
Ins8-120	1822,8	1822,8	1822,8	1768,2	1768,2	1768,2	1959,6	1987,4

Tabela 5 – Média dos resultados dos algoritmos VND's com 3 buscas locais considerando tempo de execução.

Por estes novos resultados obtidos, observa-se que os valores praticamente não se alteraram, o que se conclui que os valores ótimos já foram alcançados.

8 – CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho partiu da necessidade de se resolver o problema de minimização do tempo de espera das embarcações no porto de Imbetiba, em Macaé - RJ. Este porto, por sua particularidade de ser pequeno e exclusivo da PETROBRAS, possui regras próprias, que impedem a comparação com estudos já realizados para outros portos. Por isso, 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.

Este problema foi associado ao problema de escalonamento de tarefas em máquinas paralelas [18, 19, 20, 21, 22], que é NP-difícil. Desta forma, uma boa estratégia para resolvê-lo é fazer o uso de metaheurísticas [23, 24, 25]. Assim, foram desenvolvidos seis algoritmos baseados na metaheurística GRASP (ver Capítulo 5). Dentre eles, o que se mostrou mais adequado ao problema foi o GRASPE1BT, seguido pelo GRASPE1BI, que utilizam, na sua etapa de construção, o método ETQM1 (descrito no Capítulo 4, Seção 4.1.2).

Com base nesses resultados, foram propostos algoritmos híbridos GRASP + VND (Capítulo 6), que, nos experimentos realizados, mostraram-se eficientes para classe do problema de escalonamento de tarefas em máquinas paralelas citada neste trabalho.

Nos experimentos feitos, os algoritmos VNDTRI, VNDRTI, VNDTIR propostos apresentaram os melhores resultados para 13 dos 16 problemas testes, não sendo superado em apenas três dos problemas testes por uma pequena diferença. Isso mostra que usando o procedimento que utiliza diferentes estruturas
de vizinhanças, como o VND, pode-se conseguir benefícios quando incorporados na metaheurística
GRASP.

Para trabalhos futuros sugerem-se as seguintes opções:

- atendimento simultâneo com duas embarcações no mesmo berço quando se

tratar de embarcações pequenas;

- atendimento simultâneo com uma embarcação ao lado da outra no mesmo berço, quando uma embarcação estiver carregando fluidos;
- parada no meio do atendimento de uma embarcação para atendimento de outra de maior prioridade; e
- aplicação prática do sistema desenvolvido.

9 – REFERÊNCIAS BIBLIOGRÁFICAS

[1] Centro de estudos da UFRJ (1997). **Descrição de análise do processo de movimentação de materiais do porto de Imbetiba.** Trabalho desenvolvido para o simulador de tempo de espera das embarcações do porto de Imbetiba – Macaé.

[2] Centro de estudos da UFRJ (1997). **Modelo de Simulação de Operações portuárias**: Descrição, Utilização e Análise. Trabalho desenvolvido para o simulador de tempo de espera das embarcações do porto de Imbetiba – Macaé.

[3] Farines, J. M.; Fraga, J. S.; Oliveira, R. S. (2000), **Sistemas de Tempo Real**, 12ª Escola de Computação, IME-USP, São Paulo-SP.

[4] August, N.; Mautor, T. (1993), **Méthode Tabou Massivement Parallèle pour le Problème d'Affectation Quadratique**, Rapport de Recherche no 2182, Institut National de Recherche em Informatique et en Automatique.

[5] Chakrapani, J.; Skorin-Kapov, J. (1992), **A Connectionist Approach to the Quadratic Assignment Problem, Computers and Operations Research**, 19: 287-295.

[6] Chakrapani, J.; Skorin-Kapov, J. (1993), **Mapping Tasks to Processors to Minimize Communication Time in a Multiprocessor System**, in "The Impact of Emerging Technologies of Computer Science and Operations Research", Kluwer Academic Publishers, 45-64.

[7] Chakrapani J.; Skorin-Kapov J., (1993), **Massively Parallel Tabu Search for the Quadratic Assignment Problem**, Annals of Operations Research, 41: 327-341.

[8] Malek, M.; Guruswamy M.; Pandya M.; Owens H. (1989), **Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem**, Annals of Operations Research, 21: 59-84.

[9] Mautor, T.; Stein, L. (1994), **Recherche Tabou Parallèle Appliquée au Probléme de Placement de Tâches**, Rapport de recherche RR-94/33, Laboratoire PRISM, Université de Versailles.

[10] Porto, S.C.S.; Ribeiro, C.C. (1996), **A Case Study on Parallel Synchronous Implementations for Tabu Search Based on Neighborhood Decomposition**,

Monografias em Ciência da Computação MCC-03/96, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[11] Taillard, E. (1991), **Robust Taboo Search for the Quadratic Assignment Problem**, Parallel Computing, 17: 443-445.

[12] Taillard, E. (1992), **Parallel Iterative Search Methods for Vehicle Routing Problems**, Technical report ORWP 92/03, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausane.

[13] Taillard, E. (1994), **Parallel Taboo Search Techniques for the Job Shop Scheduling Problem**, ORSA Journal on Computing, 6: 108-117.

[14] Rassai, M. (1993), **Parallélisation d'une Méthode Approchée pour la Résolution du Problème d'Affectation Quadratique**, Mémoire d'engénieur IIE, Conservatoire National des Arts et Métiers, Institut d'Informatique d'Entreprise, Evry.

[15] Lee Y.H.; Bhaskaram, Pinedo, K.; M. (1997), **A heuristic to minimize the total weighted tardines with sequence-dependent setup times**, IIE Transactions, Vol.29, 1: 45-52.

[16] Mazzine, R. (1998), **Metaheurísticas Populacionais para o Problema de Programação de Tarefas em Máquinas Paralelas com Tempos de Preparação e Datas de Entrega**, Tese de doutorado em Engenharia Elétrica e de Computação, Universidade Estadual de Campinas – UNICAMP.

[17] Moccellin, J. V. (1994), **Comparison of Neighbourhood Search Heuristics for the Flow Shop Sequencing Problem**, Proceedings of the Fourth International Workshop on Project Management and Scheduling, 4: 228-231, Leuven-Belgium.

[18] Mendes, A.; Müller, F. M.; França, P. M.; Moscato, P. (2002), **Comparing Meta-Heuristic Approaches For Parallel Machine Scheduling Problems**, Production Planning & Control 13(2):143-154.

[19] Dearing, P. M.; Henderson, R. A. (1984), **Assigning looms in a textile weaving operation with changeover limitations**, Production and Inventory Management, 25: 23-31.

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

[21] França, P. M.; Gendreau, M.; Laport, G.; Muller, F. (1996), **A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times**, International Journal of Production Economics, 43: 79-89.

[22] Arroyo, J. E. C.; 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, 1: 1-11.

[23] Drummond, L. M. A.; Ochi, L. S.; Vianna, D. S. (2001), **An asynchronous parallel metaheuristic for the period vehicle routing problem**, Future Generation Computer Systems Journal, 17(4): 397-386.

[24] Ochi, L. S.; Drummond, L. M. A.; Victor, A. O.; Vianna, D. S. (1998), **A parallel evolutionary algorithm for solving the vehicle routing problem with heterogeneous fleet**, Future Generation Computer Systems, 14: 285-292.

[25] Vianna, D. S.; Ochi; L. S.; 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: 216:225.

[26] Feo, T.A.; Resende, M.G.C. (1995), **Greedy randomized adaptive search procedures. J. of Global Optimization**, 6:109-133.