VICTOR CAVINATO MOURA

PROGRAMAÇÃO DE FROTA DE EMBARCAÇÕES DE LANÇAMENTO DE DUTOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia

São Paulo
2012

VICTOR CAVINATO MOURA

PROGRAMAÇÃO DE FROTA DE EMBARCAÇÕES DE LANÇAMENTO DE DUTOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia

Orientador:
Prof. Dr. André Bergsten Mendes

São Paulo
2012

FICHA CATALOGRÁFICA

Moura, Victor Cavinato
Programação de frota de embarcações de lançamento de dutos / V.C. Moura. -- São Paulo, 2012.
73 p.

Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Naval e Oceânica.

1. Embarcações de apoio 2. Programação matemática 3. Programação heurística I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Naval e Oceânica II. t.

À minha família, amigos, colegas, professores e orientador pelo apoio, incentivo, companheirismo e amizade.

AGRADECIMENTOS

Gostaria de agradecer a todas as pessoas que me ajudaram ao longo da jornada para a realização deste trabalho.

Aos meus pais, Luiz Eduardo e Marta, agradeço por todo o investimento feito em minha educação, esforço, apoio e incentivo para que eu pudesse continuar estudando.

Ao orientador e amigo Prof. Dr. André Bergsten Mendes pela oportunidade, incentivo, apoio e toda contribuição dada para realização deste trabalho.

A todos os professores que contribuíram, especialmente ao Prof. Dr. Marco Antonio Brinati e Prof. Dr. Cláudio Barbieri da Cunha, pelas críticas e conselhos, muito úteis a elaboração final deste trabalho.

Aos funcionários da secretaria do PNV por toda ajuda prestada.

A Hamburg Süd, em especial aos colegas do departamento multimodal, onde trabalhei no último ano do mestrado, por todo apoio e flexibilidade dada, para que fosse possível conciliar minha vida profissional e acadêmica.

RESUMO

A presente pesquisa considera o problema de programação de uma frota de embarcações de lançamentos de dutos, conhecidas como “Pipe Layer Support Vessel” (PLSVs), as quais fazem parte da frota de apoio marítimo de uma operação “offshore”. As embarcações do tipo PLSVs são responsáveis pelas tarefas de lançamento de dutos submarinos, que escoam a produção dos poços de petróleo, e pela interligação destes dutos à infraestrutura submarina. A programação da frota deve atender uma demanda de serviço conhecida, em um horizonte de médio prazo, respeitando restrições operacionais, visando minimizar o atraso ponderado total das tarefas ou evitar que existam atrasos. Foi desenvolvido um método para estimar o valor da solução ótima do problema, baseado na técnica de relaxação lagrangiana, e um conjunto de heurísticas para gerar soluções viáveis para o problema.

Palavras-chave: Programação de veículos, Apoio marítimo “offshore”, Programação Linear Inteira Mista, Relaxação Lagrangiana, Heurísticas

ABSTRACT

This research considers the problem of scheduling a fleet of specialized vessels used for launching pipes and connecting them to the subsea infrastructure, in an offshore oil production environment. The Pipe Layer Support Vessels (PLSV) must be scheduled such that the demand is fully attended within the planning horizon, observing other operational constraints, with the purpose of minimizing the total weighted tardiness. The solution method is based on constructive and local search heuristics. Bounds on the optimal solution were derived by a lagrangean relaxation algorithm.

Key-words: Vehicle scheduling, Offshore support vessels, Mixed Integer Linear Programming, Lagrangean Relaxation, Heuristics

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

1 INTRODUÇÃO

A finalidade deste capítulo introdutório é trazer informações a respeito do problema abordado nesta pesquisa, intitulado PROGRAMAÇÃO DE FROTA DE EMBARCAÇÕES DE LANÇAMENTO DE DUTOS, definir os objetivos que se pretende alcançar, a metodologia utilizada e o encaminhamento proposto.

As embarcações do tipo “Pipe Layer Support Vessels” (PLSVs) são responsáveis pelo lançamento de dutos que transportam petróleo entre os poços dos quais são extraídos a uma unidade estacionária de produção. Isto é usualmente feito por meio de um “manifold” submarino, que consiste em uma caixa de conexão que recebe a produção de vários poços, a partir do qual o óleo é escoado à plataforma. Os PLSVs realizam o processo de interligação dos poços aos manifolds, bem como na manutenção deste sistema.

Com uma grande capacidade de armazenamento de linhas (ver figura 1.1) e de operar continuamente até em condições ambientais adversas, os PLSVs são embarcações críticas na atividade de exploração de petróleo “offshore”, devido à baixa disponibilidade no mercado, e ao alto custo de afretamento e operação.

Pretende-se, nesta pesquisa, otimizar a utilização de uma frota dedicada de embarcações de lançamento de dutos, ao longo de um horizonte de planejamento pré definido, para atender uma demanda conhecida, de forma a minimizar os custos de oportunidade associados aos atrasos na realização das tarefas.

Para atingir este objetivo, o problema será representado por um modelo matemático de programação linear inteira mista e, em seguida, resolvido de forma aproximada por um conjunto de heurísticas. O ponto de partida será uma heurística construtiva capaz de calcular um limitante

superior para o problema. Em seguida, uma heurística lagrangiana será aplicada para estimar os limitantes inferiores. Um mecanismo para integrar as duas heurísticas se utilizará de uma busca local. Esta, primeiramente, irá gerar uma solução viável a partir da solução dual e, em seguida, procurará fazer um refinamento da mesma visando encontrar um limitante superior melhor.

Para avaliar a eficácia do método de solução proposto um conjunto de instâncias de testes será desenvolvido e testado.

Figura 1.1- Embarcação tipo Pipe Layer Supply Vessel (fonte: **www.maritimejournal.com**)

1.1 Detalhamento do Problema

Diversas são as etapas de preparação de um campo de petróleo para que o mesmo entre em operação (Mendes, 2007). As embarcações de lançamento de dutos atuam na última etapa, a qual precede o início da

produção de um poço. Isto implica ter a unidade estacionária de produção já instalada – ou seja, posicionada, ancorada e parcialmente conectada aos seus sistemas produtivos.

No planejamento anual de uma empresa de petróleo é comum a criação de metas de produção, metas estas que devem ser atingidas ao longo do ano, por meio de entregas intermediárias. Neste cenário, fixa-se um horizonte de planejamento (por exemplo, um período de 4 ou 6 meses), e é necessário determinar a sequência de tarefas de lançamento de dutos e interligações que serão realizadas, levando em consideração a demanda, a frota disponível, o local e a duração das tarefas (que podem durar até 60 dias), o potencial de produção associado a cada tarefa, e o instante a partir do qual a tarefa poderia ser iniciada.

Sabe-se que a conclusão de cada tarefa gera um acréscimo nos níveis de produção da empresa, de forma que, a postergação do início de um serviço de interligação traz como consequência uma perda financeira pelo adiamento do retorno sobre o investimento realizado. Busca-se, desta maneira, priorizar as tarefas que proporcionam maiores acréscimos nos níveis de produção, desde que as restrições operacionais sejam respeitadas.

As embarcações do tipo PLSV são as responsáveis pela instalação destas linhas. Além de poder armazená-las, essas embarcações também podem efetuar a conexão de segmentos menores para, em seguida, efetuar o lançamento das linhas. Trata-se de embarcações caras, mas indispensáveis no cenário de exploração e produção “offshore”. A operação otimizada desse recurso crítico significa redução significativa dos custos e aumento da produção já a curto e médio prazo.

As tarefas de instalação e interligação são usualmente compostas de um conjunto de atividades que devem ser feitas sequencialmente. É imprescindível que haja, em primeiro lugar, o carregamento das linhas no

porto, juntamente com todos os materiais pertinentes, como conectores, ganchos, manilhas, etc. Em seguida a embarcação se dirige para o local previsto e inicia o processo de preparação, conexão e disposição da linha no leito marítimo, unindo um poço a um “manifold” submarino, ou unindo um duto rígido a um duto flexível. Pode ser necessário ir e voltar do porto repetido diversas vezes em função do comprimento da linha a ser lançada. Por último, os materiais não utilizados são devolvidos a uma base operacional.

A programação das atividades que compõem uma tarefa de interligação não é o escopo desse trabalho. É assumido que o setor de engenharia submarina da empresa petrolífera definiu previamente a melhor forma de instalar um duto ou interligar um poço, e especificou a duração mínima possível da interligação.

O tempo que cada embarcação, que compõe a frota de PLSVs da empresa, leva para executar uma interligação varia de tarefa para tarefa. É assumido que qualquer embarcação da frota seja capaz de realizar qualquer uma das tarefas previstas, observando que cada embarcação deverá executar uma única tarefa por vez, não sendo possível que mais de uma embarcação execute a mesma tarefa com o objetivo de antecipar o seu término, mesmo que existam embarcações ociosas. Além disso, se uma tarefa for iniciada, a mesma deverá ser finalizada sem a possibilidade de interromper o serviço para retomá-lo mais tarde.

Visto que uma embarcação PLSV, durante a execução de uma tarefa, se desloca constantemente até o porto e lá permanece até que os materiais sejam carregados, será admitido que o abastecimento da embarcação e a troca da tripulação sejam realizados nesses momentos em que a embarcação está atracada no porto. Além disso, a duração da execução de cada tarefa incluirá os tempos médios de porto, bem como os tempos médios de navegação entre o porto e o local da execução da tarefa, além do tempo necessário para execução do serviço. Tendo em

vista que uma tarefa sempre inicia e termina no porto, o tempo de “set-up” ou deslocamento entre duas tarefas consecutivas é zero.

Outro aspecto de fundamental importância é a data mais cedo em que a tarefa poderá ser iniciada. Esta data varia de acordo com as datas de disponibilização de material nos portos, e conforme as concessões de licenças emitidas por setores internos e externos à empresa.

Considerando os aspectos acima apontados, o problema a ser resolvido passa a ser a determinação da sequência das tarefas a serem realizadas pelas embarcações, de forma que a penalização econômico-financeira global seja minimizada. Impõe-se ainda que toda a demanda seja atendida dentro do horizonte de planejamento estipulado. A sequência das tarefas pode ser indicada pelo instante de início de cada tarefa em uma única embarcação.

1.2 **Motivação do Tema**

Soluções práticas para problemas de programação de recursos, em que estes são meramente alocados às tarefas sem a preocupação de otimizar uma função de mérito são encontradas desde as primeiras atividades exercidas pelo homem. Atualmente, o aumento da competição entre as empresas, aliada às questões de responsabilidade social e consciência ambiental, impõe a busca inexorável na melhoria dos processos e na redução dos custos envolvidos na produção e transporte de matérias primas, bens intermediários e bens finais. Isto justifica a busca por soluções ótimas para os problemas de programação de operações.

O problema descrito é de interesse prático por ser um problema real que afeta o desempenho e a continuidade das atividades de preparação dos poços para etapa produtiva. Por se tratar de uma pesquisa voltada para a resolução de instâncias de médio a grande porte,

o desafio é ainda maior, por exigir abordagens de decomposição e técnicas elaboradas de solução de problemas (Wolsey, 1998).

Além disso, tendo em vista o aquecimento do setor de petróleo, a procura por estas embarcações é extremamente elevada, algo que é refletido nas taxas diárias de afretamento destes navios. Portanto, é imprescindível que as embarcações sejam empregadas da melhor maneira possível, de forma a justificar os elevados investimentos em frota.

1.3 **Metodologia**

A solução de problemas complexos na área de planejamento de sistemas de operações é objeto central de estudo da área do conhecimento denominada de Pesquisa Operacional. É uma ciência que aplica o método científico para resolução dos mais diversos problemas operacionais, tendo como fundamento teórico os conhecimentos advindos das áreas da matemática aplicada, ciência da computação e estatística.

De acordo com Bertrand; Fransoo (2002), a Pesquisa Operacional (Operations Research) como ciência se propõe a resolver problemas na área de operações, usando os passos identificados no ciclo descrito em sua metodologia, como pode ser visto na Figura 1.2. O problema real, após a etapa de abstração, é convertido em um modelo conceitual, que traz consigo a especificação de sua extensão ou abrangência. Em seguida, a fase de modelagem matemática leva à definição das relações causais entre as variáveis do problema, tendo como produto um modelo quantitativo. O próximo passo consiste em resolver o problema representado pelo modelo matemático. Por último, vem a fase de implementação da solução.

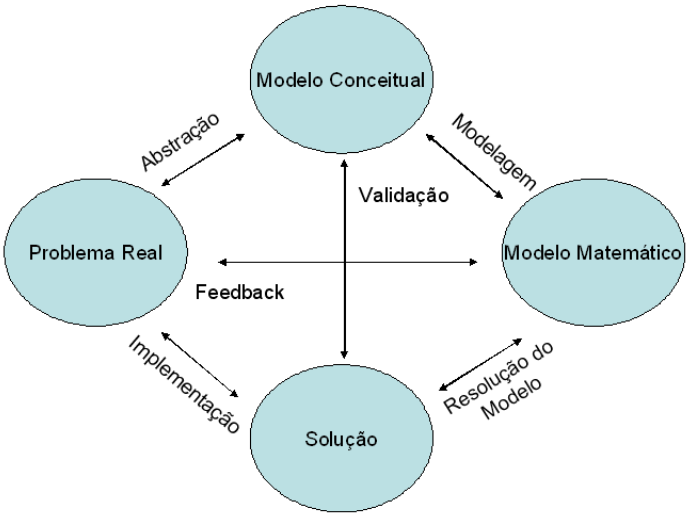

Figura 1.2 - Metodologia para resolução de problemas de pesquisa operacional, adaptado de Bertrand, Fransoo (2002)

As etapas de abstração, modelagem matemática e resolução do modelo matemático serão desenvolvidas no escopo desta pesquisa.

1.4 **Objetivos**

É objetivo desta pesquisa a resolução do problema apresentado de programação de embarcações de lançamento de dutos, para instâncias com o mesmo grau de complexidade dos problemas reais. Isto é refletido no porte dos problemas resolvidos, os quais variam de 50 tarefas e 8 embarcações, até 100 tarefas e 10 embarcações.

A despeito da existência dos pacotes comerciais de programação linear inteira mista, baseados no algoritmo de enumeração implícita “branch and bound”, constitui objetivo desta pesquisa a resolução do problema de forma aproximada por meio de métodos heurísticos.

Isto será feito calculando-se, iterativamente, os limitantes superiores e inferiores da solução, até que ambos convirjam para o

mesmo valor, atingindo o critério de otimalidade, ou que o critério de parada por número de iterações sem melhoria seja atingido.

Em relação aos limitantes superiores, Reeves (1993) define heurística como uma técnica que busca boas soluções (próximas da ótima) com um custo operacional razoável, não sendo capaz, contudo, de afirmar quão próximo uma solução factível está da solução ótima.

Para avaliar a qualidade das soluções geradas pelo método heurístico é necessário estimar o valor da solução ótima do problema. Isso será feito com o uso da relaxação lagrangiana, uma técnica que decompõe o problema, permitindo a violação de uma ou mais restrições que foram relaxadas mediante a penalização na função objetivo, como explicado em Wolsey (1998).

O problema em questão será, inicialmente, modelado segundo a estratégia de discretização temporal proposta por Luh et al. (1990) e Akker et a. (2001), no qual as variáveis de decisão determinam a data de início de cada tarefa, observando as diversas restrições.

Por fim, diversos testes computacionais serão conduzidos com o propósito de avaliar a eficácia do processo de solução proposto.

1.5 **Organização do Texto**

O presente texto está organizado da seguinte maneira:

- No capítulo 2 *Revisão Bibliográfica,* é feito o levantamento de alguns trabalhos relacionados à área de transporte marítimo, problemas de programação de veículos e são apresentados os principais métodos utilizados na resolução de problemas de programação de tarefas.

- No capítulo 3 *Modelagem Matemática* é apresentada uma formulação matemática, indexada ao tempo, para representar o problema.
- No capítulo 4 *Métodos de Solução* são apresentados a heurística construtiva, o método de relaxação lagrangiana, a heurística que torna uma solução inviável em viável e a heurística de busca local.
- No capítulo 5 *Aplicação dos Métodos de Solução* são mostrados e discutidos os resultados obtidos com o algoritmo para diversas instâncias de teste.
- No capítulo 6 *Conclusões e Recomendações* são expostas as conclusões da pesquisa além das considerações finais a respeito dos resultados obtidos.
- No capítulo 7 *Referências Bibliográficas* será apresentada a bibliografia consultada.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão revisados alguns trabalhos relacionados à indústria de transporte marítimo, dando ênfase à modalidade de operação do tipo “frota industrial”. Devido às semelhanças encontradas no problema de sequenciamento de tarefas para um conjunto de embarcações e os problemas de programação em máquinas paralelas, será feita uma revisão da literatura referente a alguns métodos de modelagem, comumente empregados nestes problemas de programação de máquinas paralelas. Por fim, será apresentada uma visão geral das técnicas de resolução de problemas de programação inteira e inteira mista, juntamente com a revisão de alguns trabalhos relacionados.

2.1 Modalidades de Transporte Marítimo

O transporte marítimo é o principal modal de transporte para as cargas provenientes de comércio internacional. De acordo com Christiansen et al. (2004), o aumento populacional, o aumento da industrialização, a ausência ou escassez de insumos de produção, a eliminação das barreiras comerciais, entre outros motivos, contribui para o crescimento contínuo deste modal de transporte.

Os problemas de planejamento de frota em transporte marítimo são usualmente divididos em: linhas regulares, linhas não regulares e frota industrial (Lawrence, 1972).

O transporte por meio de linhas regulares é representado principalmente pelas empresas de navegação que transportam contêineres e carga geral, de clientes que não têm carga suficiente para justificar o afretamento de um navio inteiro. Esta operação tem como característica a oferta de serviço entre diferentes portos do mundo,

seguindo roteiros previamente estabelecidos, que são divulgados para o mercado, contendo as datas de chegada e saída dos portos que fazem parte da rota.

O transporte por meio de linhas não regulares, por sua vez, é caracterizado pelo atendimento de demandas específicas de um pequeno grupo ou de uma única empresa que possuem grande volume de carga de commodities, como grãos minérios e petróleo. As principais decisões que os usuários deste tipo de transporte têm de tomar se baseiam na escolha da composição da frota bem como da do tipo de contrato a ser feito para afretamento dessas embarcações.

Já a modalidade frota industrial é comum em grandes grupos empresariais que possuem ativos nas mais diversas etapas do processo produtivo, sendo proprietári0s das embarcações, ou afretando-as por longos períodos. Estas são utilizadas para atender suas demandas por transporte de insumos ou produtos acabados.

Em muitos casos a diferenciação desses problemas não é muito clara. Uma embarcação pode facilmente mudar o regime de operação ou então o dono da frota pode ter embarcações operando simultaneamente sob regimes diferentes (Ronen, 1983).

Neste trabalho, está sendo considerada a frota de apoio marítimo “offshore” que uma empresa de petróleo dispõe a qual, entre outros tipos de embarcações, é composta por PLSVs próprios ou afretados, mediante contratos de longo prazo. Pelo fato das embarcações do tipo PLSV serem embarcações especializadas, não existe no mercado de linhas não regulares disponibilidade desse tipo de embarcação, ficando a própria empresa responsável por garantir uma composição de frota que atenda às suas necessidades a longo prazo, como foi destacado por Christiansen et al. (2004).

Mendes (2007) mostra que as operações de exploração marítima offshore podem ser caracterizadas como operação de uma frota industrial. Assim sendo, é comum a ocorrência de problemas de roteirização e programação de veículos visando minimizar os custos operacionais ou maximizar as receitas advindas das tarefas realizadas por essas embarcações especializadas. Devido ao horizonte de planejamento de curto a médio prazo em que estes problemas estão inseridos, considera-se que a composição da frota está pré-determinada, não cabendo ao modelo determinar a composição ideal da mesma, mas apenas otimizar o seu uso.

2.2 Problemas de Programação de Veículos e Problemas de Programação de Máquinas Paralelas

Os problemas de roteirização e programação de veículos guardam várias semelhanças e algumas diferenças em relação aos problemas de programação de tarefas em máquinas paralelas. Estudos a respeito das semelhanças e diferenças mais comuns entre os modelos foram feitos em Beck et al. (2003) e Kouki et al. (2007). Considerando o problema proposto nesta pesquisa, em função das premissas assumidas, o mesmo pode ser caracterizado como um problema de programação de tarefas em máquinas (no caso, embarcações) paralelas.

Um sistema de produção em máquinas paralelas pode ser visto como um caso particular dos problemas de processamento em múltiplas máquinas. Cheng; Sin (1990) definiram-no como um sistema onde uma tarefa pode ser processada em qualquer máquina livre; além disso, cada tarefa que terminou de ser processada deixa o sistema liberando o recurso que estava utilizando. Neste trabalho é feito um levantamento a respeito dos diferentes tipos de problemas de programação de máquinas, de acordo com a característica das tarefas (tempo de processamento, data de entrega, restrições de precedência) e critérios de performance

(atraso, atraso ponderado, tempo total para término de todas as tarefas). Também são citadas as quatro principais classes de métodos de solução para problemas de otimização combinatória, nas quais os problemas de programação em máquinas paralelas se enquadram: enumeração completa, algoritmos em tempo exponencial, algoritmos em tempo polinomial e algoritmos aproximados. Além disso, é feita uma extensa revisão bibliográfica dos trabalhos de programação em máquinas paralelas, dividindo-os de acordo com o critério de apuração da função objetivo.

Em Chan et. al (1997) é analisada a resolução de problemas de programação em máquinas paralelas por meio da relaxação linear do problema formulado segundo o problema de partição de conjuntos. De acordo com os autores, o sucesso do método se apoia sobre dois pilares: a habilidade para resolver o problema linear relaxado, comumente feito por meio do método de geração de colunas e o fato dessa relaxação gerar bons limitantes inferiores. É citado também o uso de heurísticas de aproximação, que ordenam as tarefas em listas, por meio dos mais diversos critérios para, em seguida, alocá-las às máquinas, de acordo com a disponibilidade das mesmas, gerando soluções inteiras a partir de soluções lineares.

Algumas formulações matemáticas, relaxações e algoritmos exatos para o problema de programação de máquinas paralelas não idênticas são apresentados em Li; Yang (2009). As heurísticas apresentadas naquele trabalho estão divididas em três grupos: algoritmos de aproximação, meta-heurísticas e outras heurísticas. Os algoritmos de aproximação são definidos como algoritmos que, partindo de soluções ótimas para problemas relaxados, são capazes de garantir uma determinada solução em tempo polinomial. As meta-heurísticas são descritas como algoritmos de busca local capazes de encontrar soluções de boa qualidade, sem garantia de otimalidade. Por fim, as heurísticas

construtivas gulosas são algoritmos que não têm performance garantida e não utilizam estratégias de busca local.

A utilização de regras de dominância para auxiliar na resolução de problemas de programação de máquinas paralelas é analisada em Jouglet; Savourey (2011). De acordo com os autores, embora tenha sido feita uma revisão bibliográfica a respeito de regras de dominância para problemas de máquinas paralelas, a introdução de datas de liberação ao problema, inviabilizava o uso das regras de dominância até então conhecidas. Neste caso, foram desenvolvidas novas regras baseadas no conceito de máquinas onde uma tarefa pode ser programada.

O crescente interesse por trabalhos na área de programação de embarcações pode ser explicado pelo alto custo de capital para aquisição de embarcações e o alto custo operacional das mesmas. Isto significa que o aumento de produtividade da frota pode levar a grandes melhorias no resultado financeiro da empresa. Além disso, o grande aumento da capacidade de processamento, o aumento de memória, e a diminuição nos custos dos computadores ao longo das últimas décadas, levaram à disseminação dos métodos de solução dos mais variados tipos para problemas de programação.

Em Unlu e Mason (2010) são propostas e avaliadas quatro formulações para o problema de máquinas paralelas sem restrição de precedência. As diferentes formulações são avaliadas quanto a suas vantagens e desvantagens em cenários cuja quantidade de máquinas disponíveis varia. Cita ainda que a formulação indexada ao tempo por eles proposta, apresenta algumas dificuldades quando do aumento do número de tarefas e do aumento do tempo de processamento dessas tarefas. Mesmo para instâncias pequenas, caso o tempo de processamento das tarefas seja alto, a quantidade de variáveis e restrições do modelo indexado ao tempo é muito grande. Apesar destas desvantagens, o artigo cita que essa formulação foi a que teve o melhor

desempenho dentre as formulações testadas e que foi a única capaz de gerar soluções ótimas para problemas de 5, 10 e 15 máquinas.

2.3 Métodos de Solução para os Problemas de Programação Inteira Mista

O problema apresentado neste trabalho consiste em determinar a sequência de atendimento das tarefas que minimiza as perdas econômicas - financeiras pelo adiamento da entrada em produção dos poços a serem interligados. A formulação matemática que será apresentada requer que os métodos de solução consigam lidar com a natureza discreta das variáveis de decisão.

Para lidar com a natureza discreta dos problemas de programação, existem disponíveis na literatura diversos tipos de abordagem entre elas: métodos exatos que determinam a solução ótima do problema, além de métodos heurísticos que fornecem soluções aproximadas para o problema. O uso dos métodos citados pode se dar de forma isolada ou de forma combinada, de acordo com as características e especificidade inerentes a cada tipo de problema.

2.3.1 Métodos Exatos

Os métodos exatos para resolução de problemas de programação inteira podem ser divididos em métodos de enumeração explícita e métodos de enumeração implícita.

O método de enumeração explícita enumera todas as possíveis soluções do problema comparando o valor da função objetivo em busca da solução ótima do problema. Os métodos de enumeração implícita dividem o problema maior em subproblemas menores e mais fáceis de

serem resolvidos para então obter a solução do problema principal Land (1960).

Dentre as estratégias de enumeração implícita comumente empregadas, pode-se mencionar o método “branch-and-bound” (Wolsey, 1998). Este consiste em resolver a versão linear relaxada do problema original e, sucessivamente, eliminar as variáveis fracionárias pela separação da região factível.

O método “branch-and-bound”, ao longo da sua execução, armazena em sua memória a melhor solução viável encontrada até o momento, além de um limitante inferior (problema de minimização) do valor da função objetivo da melhor solução do problema. As informações armazenadas são utilizadas para descartar o espaço de busca fazendo com que o algoritmo convirja mais rapidamente. Quanto menor for a diferença entre o limitante superior e o limitante inferior, mais próximo o algoritmo está de determinar a solução ótima para o problema que está sendo resolvido. Em Cordeau et al. (2006) é feita uma crítica ao método, pois caso a qualidade do limitante inferior não seja boa, o método é pouco eficaz.

A forma como a busca na árvore “branch-and-bound” é feita varia, podendo ser em profundidade, em largura, ou utilizando algum critério híbrido. A melhor estratégia de busca pode variar de acordo com as características do problema e de acordo com as limitações computacionais do equipamento onde o algoritmo está sendo processado. A vantagem da estratégia da busca em profundidade está no fato dela manter menor número de nós ativos, exigindo menor limite de memória computacional, com a desvantagem de na maioria dos casos gerarem árvores com maior número de nós, exigindo maior tempo computacional para encontrar a resolução do problema.

A técnica dos planos de corte também é muito utilizada na resolução de problemas de programação inteira e consiste em gerar inequações válidas com o objetivo de reduzir o espaço de busca do problema linear relaxado sem, no entanto, excluir nenhuma solução inteira viável do problema original. Mais detalhes de como as inequações são geradas, podem ser encontrados em Wolsey (1998).

Outro método exato para resolução de problemas de programação inteira mista, que merece atenção na hora de resolver problemas difíceis de programação inteira mista é a técnica do “branch-and-cut”. Este método é uma extensão algoritmo “branch-and-bound”, em que a resolução da relaxação linear em cada nó da árvore é acompanhada com a adição de cortes.

2.3.2 Métodos Heurísticos

Embora não exista a garantia de que um método heurístico possa encontrar a solução ótima de um problema, ou mesmo que essa solução possa ser reconhecida caso seja alcançada, o uso desse tipo de método é justificado pelo fato de poder gerar soluções de boa qualidade, com custo computacional razoável.

Quando se depara com problemas de grande porte, para os quais os métodos exatos são proibitivos, quer seja por limitação de memória, por limitações de tempo de processamento, ou mesmo tempo hábil para desenvolvimento de um método exato, por mudanças constantes nos parâmetros de entrada do problema, os métodos heurísticos podem ser a melhor escolha para a resolução desse tipo de problema.

Outra vantagem dos métodos heurísticos está na flexibilidade que os mesmos oferecem. De acordo com análises de Cunha (1997, 2000), a respeito dos aspectos práticos da aplicação de modelos de roteirização

para problemas de programação de veículos, os métodos heurísticos se apoiam em uma abordagem intuitiva, para tomar proveito da estrutura de um problema específico de um modo inteligente, visando à obtenção de soluções adequadas. Ainda de acordo com Cunha (2000) o fato de as heurísticas serem desenvolvidas para tratamento de um problema em particular, faz com que as mesmas não apresentem muita robustez uma vez que, quando submetidas a problemas com algumas modificações, os resultados alcançados não têm garantia de qualidade, podendo piorar consideravelmente.

Heurísticas construtivas são métodos, geralmente gulosos, que constroem soluções a partir de uma técnica de adição. A cada iteração é agregado um elemento a uma solução parcial, de forma que esta construção siga um processo contínuo e gradativo, e que ao final gere uma solução para o problema apresentado. A maioria das heurísticas construtivas tem aplicação quase que exclusiva ao problema específico para o qual elas foram concebidas para resolver. Em muitos casos é possível trabalhar em conjunto com outros métodos de solução, gerando soluções viáveis que irão alimentar outros métodos, quer sejam exatos, quer sejam heurísticos.

Heurísticas de Busca Local são heurísticas criadas para explorar as vizinhanças das soluções, visando melhorar o valor da função objetivo, por meio de modificações em uma ou em um conjunto de soluções já definidas. Para construção das vizinhanças é necessário definir regras para alteração de soluções e geração de novas soluções viáveis para o problema em questão.

Em geral, as Heurísticas de Busca Local são empregadas a partir de estruturas de busca conhecidas como Metaheurísticas. Estas estruturas possuem mecanismos para direcionar a busca em regiões distintas do espaço de solução e impedir que o algoritmo interrompa prematuramente ao atingir um ótimo local. As Metaheurísticas combinam

etapas de construção, busca local e diversificação. Algumas exploram a busca em torno de uma única solução, enquanto que outras trabalham com um conjunto de soluções.

Dentre as principais Metaheurísticas encontradas na literatura, podem ser citadas: Busca Tabu, Algoritmo Genético, GRASP, Scatter Search, Simulated Annealing, VNS (Variable Neighborhood Search), entre outros. Em Aarts (1997) podem ser encontradas mais detalhes a respeito dos métodos da Busca Tabu e Algoritmo Genético.

Em Hansen e Mladenovic (2003) são mostrados, de forma bem detalhada, os princípios do funcionamento do método VNS assim como suas variações. O VNS geral é apresentado, mostrando como funciona a sistemática de geração de vizinhanças e aplicação de técnicas de busca local sobre essas vizinhanças. Além disso, são discutidas questões que levam ao aumento de performance do método, como por exemplo a geração de soluções inicias de boa qualidade, a escolha dos tipos de vizinhança a serem gerados e os tipos de busca local a serem realizadas. No final é feita uma lista contendo as propriedades desejáveis para que um algoritmo VNS possa ter um bom desempenho.

Como nem sempre é possível estabelecer quão próxima a solução obtida em um método heurístico está da solução ótima de um problema, é importante que se possa encontrar um limitante inferior do valor da solução ótima de um problema para uma dada instância. A solução exata pode ser obtida, como já foi dito anteriormente, com uso do método do branch and bound, embora não seja o único método que possa ser utilizado para tanto. Se for possível resolver de forma exata o problema original, relaxado de uma ou mais restrições que dificultam sua resolução, limitantes do valor da solução ótima do problema original poderão ser obtidos.

2.3.3 Métodos de Relaxação

Dentre as técnicas de relaxação comumente utilizadas pode ser citada a Relaxação Linear que modifica o problema original, retirando das restrições de integralidade e resolvendo o problema relaxado como se fosse um problema linear (Beasley, 1983).

Outro método de relaxação muito utilizado para gerar limitantes das soluções ótimas para os problemas é o Método da Relaxação Lagrangiana que consiste em retirar uma ou mais restrições que estejam dificultando a resolução de um problema de programação inteira ou inteira mista, e inserir essas restrições na função objetivo, ponderadas por multiplicadores de Lagrange. A resolução desse problema modificado de forma exata gera uma estimativa do valor ótimo para a função objetivo do problema original. Muitos problemas de otimização combinatória podem ser divididos em um subproblema mais fácil de resolver, e uma restrição que dificulta a aplicação dos métodos de solução. Para esses casos, o método da relaxação lagrangiana funciona muito bem. Além disso, experiências práticas mostram que as soluções indicadas pelo método geram boas limitantes para o valor ótimo da solução do problema (Reeves, 1993).

No trabalho de Luh et al. (1990) é proposto um método de solução, para o problema de máquinas paralelas que devem executar tarefas de diferentes durações, com diferentes importâncias e diferentes instantes de liberação. A função de mérito que se pretende otimizar no artigo de Luh et al. (1990) é o atraso total ponderado. Utilizando o método da relaxação lagrangiana, obteve-se um algoritmo eficiente para estimar o valor da solução ótima que em média gerava estimativas a 1% do ótimo. A metodologia utilizada para atualizar os multiplicadores de lagrange se baseava no método do subgradiente. As soluções geradas para o problema relaxado estavam muito próximas de soluções viáveis. Um

algoritmo guloso era responsável por transformar soluções inviáveis em viáveis.

Em Espejo e Galvão (2002) são avaliados três métodos de relaxação: relaxação lagrangiana, relaxação surrogate e relaxação combinada lagrangiana-surrogate. O método da relaxação surrogate, exposto pelos autores consiste em combinar algumas restrições do problema original a uma única restrição, denominada restrição surrogate. De acordo com os autores, Glover (1975) e Greeberg & Pierskalla (1970) foram os primeiros a demonstrar que o gap dual do enfoque surrogate é necessariamente menor ou igual ao gap dual do enfoque lagrangiano. A desvantagem da relaxação surrogate em relação a relaxação lagrangiana fica por conta do fato de as aplicações serem mais restritas, o que pode ser explicado pelo fato de que os problemas obtidos com a aplicação da relaxação surrogate serem de difícil resolução.

3 MODELAGEM MATEMÁTICA

Este capítulo apresenta o modelo matemático utilizado para representar o problema de “PROGRAMAÇÃO DE FROTA DE EMBARCAÇÕES DE LANÇAMENTO DE DUTOS”. Inicialmente serão estabelecidas as hipóteses e premissas do problema em questão; após uma breve discussão da estratégia de modelagem, os parâmetros, as variáveis de decisão, a função objetivo e as restrições serão apresentadas.

3.1 Premissas do Problema

Com base na descrição do problema apresentada no capítulo inicial desta pesquisa, o problema em questão será resolvido tendo como base as seguintes premissas:

- A demanda é totalmente conhecida a priori;
- Cada tarefa possui um tempo específico de execução;
- Cada tarefa possui uma data de liberação, antes da qual não é permitido o seu início;
- Cada tarefa tem uma data de entrega prevista;
- Tarefas finalizadas após a data de entrega prevista estarão sujeitas a uma penalização que incide para cada dia de atraso, acumulando de forma linear na função objetivo;
- Uma tarefa, após ser iniciada, não é interrompida;
- Cada tarefa é realizada por apenas uma única embarcação;
- Uma embarcação não pode executar mais de uma tarefa ao mesmo tempo;
- As tarefas consideradas neste trabalho são independentes entre si, não existindo quaisquer restrições de precedência entre as mesmas;

- Adotou-se a hipótese simplificadora de que frota é homogênea, e qualquer embarcação está apta para realizar todas as tarefas;
- O tempo de deslocamento de uma embarcação entre tarefas é nulo, uma vez que a tarefa inicia e termina em um porto.

3.2 **Estratégia da Modelagem**

Conforme apresentado na metodologia de Bertrand e Fransoo (2002), a modelagem matemática consiste da proposição da relação causal entre as variáveis identificadas que, no caso desta pesquisa, referem-se à definição do instante de início de cada tarefa.

A formulação adotada para representar o problema é indexada ao tempo, adaptada de Van den Akker et al. (2000). Nesta discretização, o horizonte de tempo é dividido em períodos, de forma que o período t inicia no instante t-1 e termina no instante t, e compreende T períodos.

As vantagens das formulações indexadas ao tempo podem ser resumidas como a facilidade com que a mesma pode ser usada para modelar diversos tipos de problemas de programação de uma única máquina ou de máquinas paralelas e, além disso, as relaxações aplicadas sobre esta formulação costumam gerar bons limitantes de solução. A principal desvantagem reside no tamanho dos problemas gerados, que exigem bastante memória e tempo de processamento. Em Unlu e Mason (2010) são avaliadas formulações distintas para os problemas de programação inteira mista de máquinas paralelas idênticas sem restrição de precedência. Os autores ressaltam o elevado consumo de memória para armazenagem de informações, mesmo para instâncias pequenas, quando o número de períodos é grande.

A discretização do tempo, que é um parâmetro contínuo, pode, eventualmente, afetar a qualidade da solução. À medida que a duração

dos eventos tem que ser aproximada por número inteiro de períodos unitários, o modelo matemático passa a representar uma versão simplificada da realidade. Se a indexação for feita considerando intervalos de tempo muito pequenos, serão necessários muitos intervalos para preencher todo o horizonte de planejamento, implicando em aumento da dificuldade computacional para resolução do problema. Por outro lado, a utilização de intervalos de tempo muito longos, pode simplificar em demasia a representação do problema fazendo com que a solução ótima do modelo não necessariamente represente a solução ótima do problema real.

3.3 **Formulação Indexada ao Tempo**

Parâmetros Referentes ao Problema:

N Conjunto de tarefas

E Conjunto de embarcações

T Horizonte de planejamento

m Número de embarcações

n Número de tarefas

Parâmetros Referentes as Tarefas e Embarcações:

c_j^t Custo de oportunidade incorrido quando a tarefa $j \in N$ é iniciada no instante t

p_j Duração da tarefa $j \in N$

r_j Data a partir da qual a tarefa $j \in N$ pode ser iniciada

Variáveis de Decisão:

$x_j^t = 1, \text{se a tarefa } j \in N \text{ inicia na data } t;\ 0, \text{em caso contrário}$

Função Objetivo:

$$\min C = \sum_{j=1}^{n} \sum_{t=r_j}^{T-p_j+1} c_j^t x_j^t \tag{3.1}$$

Sujeito a:

$$\sum_{t=r_j}^{T-p_j+1} x_j^t = 1 \qquad j \in N \tag{3.2}$$

$$\sum_{\substack{j=1: \\ r_j \le t}}^{n} \sum_{s=\max(r_j, t-p_j+1)}^{\min(t, T-p_j+1)} x_j^s \le m \quad t = 1, \dots, T \tag{3.3}$$

$$x_j^t \in \{0,1\} \qquad j \in N, \quad t = 1, \dots, T \tag{3.4}$$

A função objetivo (3.1) representa o custo total de oportunidade incorrido devido ao início das tarefas nos respectivos instantes, de acordo com as variáveis de decisão x_j^t e a matriz de custos de oportunidade c_j^t. A restrição (3.2) garante que todas as tarefas sejam iniciadas uma única vez. A restrição (3.3) garante que o número de tarefas sendo executadas em um dado momento não ultrapasse o número de embarcações disponível. Por fim, a restrição (3.4) indica que as variáveis de decisão x_j^t são binárias.

4 MÉTODOS DE SOLUÇÃO

Neste capítulo serão apresentados os métodos utilizados para solução do problema de PROGRAMAÇÃO DE FROTA DE EMBARCAÇÕES DE LANÇAMENTO DE DUTOS.

4.1 Considerações Iniciais

Na revisão bibliográfica foram apresentados diversos métodos utilizados na resolução de problemas de programação de recursos. Nesta pesquisa optou-se por resolver o problema de programação de frota por meio de heurísticas. O desafio desta abordagem está na necessidade de desenvolver limitantes inferiores do valor da função objetivo, e encontrar limitantes superiores de boa qualidade e, se possível, a solução ótima.

Tendo em vista que os limitantes inferiores são obtidos por meio de soluções inviáveis, mas cujo valor da função objetivo pode, eventualmente, estar próximo do valor da solução ótima, procurou-se investigar se na vizinhança de uma solução inviável poderia ser encontrada uma solução viável de boa qualidade.

O método de solução proposto consiste, em primeiro lugar, gerar uma solução inicial com o suporte de uma heurística construtiva. Esta heurística se baseia em uma abordagem intuitiva, visando explorar de forma apropriada as características do problema. A heurística implementada atribui prioridades às tarefas demandadas, de forma que as mesmas possam ser ordenadas em uma fila. Uma a uma, as tarefas que têm maior prioridade são alocadas na embarcação que está disponível mais cedo, até que todas as tarefas estejam alocadas.

A geração de uma solução factível não só constitui um limitante superior para o problema, como também é um importante parâmetro de entrada no método da relaxação lagrangiana. Embora não seja possível garantir que a heurística construtiva forneça sempre uma solução factível, para todas as instâncias de teste o algoritmo teve êxito na busca de uma solução viável para o problema.

A relaxação proposta executa duas funções importantes dentro do método de solução proposto. A primeira delas é prover um limitante inferior do valor da função objetivo do problema, para que a qualidade das soluções primais possa ser avaliada. A segunda delas é fornecer vizinhanças novas, a cada iteração, para alimentar a heurística de busca local.

A heurística de busca local foi estruturada a partir de uma adaptação do método VNS ("variable neighborhood search"), proposto por Hansen (2003). A geração de vizinhanças é feita a partir de uma heurística que transforma as soluções inviáveis, geradas pelo método da relaxação lagrangiana, em soluções viáveis. Os três métodos de busca local são então, aplicados sobre essas soluções viáveis "adaptadas", fornecidas a cada iteração.

Quais buscas locais utilizar e quantas vezes cada um desses tipos de busca realizar sobre as novas vizinhanças são parâmetros de calibragem do algoritmo. Informações a respeito destes parâmetros de calibragem, dos critérios de parada e mais detalhes a respeito das heurísticas e método de relaxação serão explicitados a seguir.

4.2 **Heurística Construtiva**

A heurística construtiva proposta nesta pesquisa é composta de uma heurística gulosa que funciona de forma recursiva, aliada a um

método de busca local que visa aprimorar a solução construída antes de iniciar a próxima etapa (relaxação lagrangiana).

Na heurística gulosa, a cada iteração uma tarefa é selecionada e alocada a uma embarcação. Para definir qual tarefa será alocada a qual embarcação em uma determinada iteração, são utilizados alguns critérios que dependem das características da tarefa (data de liberação, data de entrega, penalidade por atraso, duração da tarefa).

O motivo de a heurística ser caracterizada como gulosa é que uma vez que uma tarefa foi escolhida e inserida em uma embarcação, a mesma não poderá ser transferida de embarcação ou de posição, independentemente das características das tarefas restantes que ainda não foram inseridas. A primeira tarefa alocada a uma embarcação será a primeira tarefa a ser executada por essa embarcação. Cada alocação subsequente em uma embarcação é feita de forma que a tarefa inserida seja executada depois das tarefas alocadas anteriormente na mesma embarcação.

Para definir qual tarefa será alocada a uma embarcação, a cada iteração são calculados os seguintes parâmetros:

- Penalização Marginal: Calcula-se qual o custo em que se incorre caso o início da tarefa seja postergado em 1 (um) intervalo de tempo em relação ao instante mais cedo disponível t, entre todas as embarcações. O instante disponível das embarcações é o instante seguinte à conclusão de todas as tarefas associadas àquela embarcação. Tarefas que ainda não foram liberadas para início no instante t, assumem penalização marginal = 0, para efeito de cálculo de prioridade.
- Folga: Calcula-se a quantidade de períodos de tempo que separam o hipotético final da tarefa, caso ela seja alocada à embarcação que está disponível mais cedo, e a data de entrega desta tarefa. Tarefas que ainda não foram liberadas

para início, assumem folga infinita para efeito de cálculo de prioridade.

Após o cálculo dos parâmetros acima explicitados, a próxima tarefa a ser alocada em uma dada iteração é definida de acordo com os critérios de maior penalização marginal, menor folga, maior peso por atraso, menor duração da tarefa. A escolha é feita de forma que, caso tenha-se empate no primeiro critério, a escolha é feita pelo segundo critério. Caso o empate persista, a tarefa é escolhida de acordo com o próximo critério e assim sucessivamente até o final de todos os critérios. Caso as tarefas tenham todos os critérios iguais, a escolha é feita pela primeira tarefa avaliada. Quanto à escolha da embarcação, será priorizada aquela que possuir o instante mais cedo de disponibilidade. Em caso de empate, será escolhida a embarcação de índice menor.

O procedimento descrito acima é repetido até que todas as tarefas demandadas estejam alocadas às embarcações. Com todas as tarefas alocadas, gera-se a primeira solução para o problema, que pode ser viável ou não, já que na heurística gulosa é possível que a programação proposta não esteja respeitando o horizonte de planejamento do problema.

Nos casos em que não for possível gerar uma solução viável, é aplicado um segundo procedimento tentativo, visando a geração de soluções que respeitem o horizonte de planejamento. Para isso foram utilizadas ideias intuitivas, comumente aplicadas em problemas de minimização de tempo total (“makespan”), para processamento de uma lista de tarefas. O procedimento consiste em listar as tarefas de acordo com a ordem decrescente dos tempos de processamento e em seguida alocá-las uma a uma na embarcação que estiver disponível mais cedo. Conforme explicado em De e Morton (1980), o uso desta técnica balanceia o tempo de trabalho dos processadores, e minimiza o efeito das últimas alocações já que estas têm menor duração.

Na fase final da heurística construtiva, é aplicada uma busca local à solução fornecida pela heurística gulosa visando garantir que esta solução, além de viável, tenha uma boa qualidade. Este valor será o limitante superior da relaxação lagrangiana. O procedimento de busca local é o mesmo empregado na busca da solução gerada no âmbito da relaxação lagrangiana, e será detalhado na seção 4.4.2.

4.3 **Relaxação Lagrangiana**

Quanto ao problema desta pesquisa, representado pelo modelo descrito no capítulo anterior, pode-se observar que a solução do mesmo é facilitada quando a restrição de capacidade (3.3) é relaxada. Na ausência de tal restrição, é possível sobrepor tarefas de forma que todas possam ser programadas sem incorrer em atraso, sendo simples a proposição de uma solução ótima para o problema.

Considerando, contudo, que é mandatório que uma solução respeite a disponibilidade de embarcações, a violação da capacidade será penalizada na função objetivo por meio dos multiplicadores de Lagrange. Um novo problema passará a ser a determinação do valor ótimo desses multiplicadores.

Como explicado resumidamente nas considerações iniciais desse capítulo, a relaxação lagrangiana tem dois papéis fundamentais no método de solução proposto. O primeiro deles é fornecer um limitante inferior e, o segundo, é alimentar a heurística de busca local com diferentes pontos de partida.

Para resolver o problema relaxado será proposta uma estratégia de solução, na qual o problema é decomposto em subproblemas menores, de fácil solução, sendo um subproblema por tarefa. Esta abordagem de

decomposição é favorecida pelo caráter aditivo da função objetivo e da restrição de capacidade.

A atualização dos multiplicadores de Lagrange será feita por meio do algoritmo do subgradiente, adaptado de Luh et al. (1990), que será introduzido em seguida.

4.3.1 Problema Relaxado (R)

Relaxando a restrição de capacidade (3.3) e penalizando-a na função objetivo com o uso dos multiplicadores de Lagrange π^t, o problema relaxado R adquire a seguinte estrutura:

$$\mathrm{R}(\pi) = \sum_{j=1}^{n} \sum_{t=r_j}^{T-p_j+1} c_j^t x_j^t + \sum_{t=1}^{T} \pi^t \left(\sum_{\substack{j=1: \\ t \geq r_j}}^{n} \sum_{s=\max(r_j, t-p_j+1)}^{\min(t, T-p_j+1)} x_j^s - m \right) =$$

$$= -\sum_{t=1}^{T} \pi^t \cdot m + \sum_{j=1}^{n} \sum_{t=r_j}^{T-p_j+1} c_j^t x_j^t + min \left(\sum_{t=1}^{T} \sum_{\substack{j=1: \\ t \geq r_j}}^{n} \sum_{s=\max(r_j, t-p_j+1)}^{\min(t, T-p_j+1)} \pi^t \cdot x_j^s \right) \quad (4.1)$$

Sujeito a: (3.2), (3.4) e

$$\pi^t \geq 0 \qquad \forall t \qquad (4.2)$$

4.3.2 Problema Dual (D)

Considerando que a penalização introduzida à função objetivo é multiplicada por um fator maior ou igual a zero, o valor da função objetivo do problema relaxado sempre será menor ou igual a função objetivo do problema original (3.1). Dado que um limitante inferior para a função objetivo é dado por R(π), a função dual está então maximizada, no espaço de solução das variáveis π^t $(t: 1, \dots, T)$, dando origem ao problema dual D.

$$D = max_{\pi \geq 0} \quad R(\pi) =$$

$$max_{\pi \geq 0} \left\{ min \left(-\sum_{t=1}^{T} \pi^t m + \sum_{j=1}^{n} \sum_{t=r_j}^{T-p_j+1} c_j^t x_j^t + \sum_{t=1}^{T} \sum_{\substack{j=1: \\ t \geq r_j}}^{n} \sum_{s=\max(r_j, t-p_j+1)}^{\min(t, T-p_j+1)} \pi^t x_j^s \right) \right\} \quad (4.3)$$

Sujeito a: (3.2), (3.4)

4.3.3 Programação das Tarefas

Em primeiro lugar, a expressão (4.3) será reescrita, decompondo a função objetivo, em termos independentes, por tarefa.

$$D = max_{\pi} \left\{ \sum_{j=1}^{N} \min D(j, \pi) - \sum_{t=1}^{T} \pi^t m \right\} \quad (4.4)$$

Em que:

$$D(j, \pi) = \sum_{t=r_j}^{T-p_j+1} c_j^t \, x_j^t + \sum_{t=r_j}^{T} \sum_{s=\max(r_j, t-p_j+1)}^{\min(t, T-p_j+1)} x_j^s \, \pi^t \quad (4.5)$$

Para um dado valor de π^t $(t: 1, \dots, T)$, a resolução de (4.4) pode ser feita resolvendo-se subproblemas (4.5) por tarefa. Cabe ressaltar que o primeiro termo de (4.4), a saber: $-\sum_{t=1}^{T} \pi^t m$, é constante.

Dado que a função $D(j,\pi)$ deve ser minimizada para que o instante de início de uma tarefa seja determinado, é necessário avaliar, dentre as datas de início $r_j \leq t \leq T - p_j + 1$, aquela que minimiza a função $D(j,\pi)$, o que pode ser feito por comparação explicita de todas as alternativas.

A título de ilustração, seguem abaixo o exemplo de uma tarefa fictícia e o subproblema a ela associado. Considere uma tarefa *j*, tal que r_j = 9; p_j = 3; T = 14. A função $D(j,\pi)$ pode ser escrita como:

$$\min D(j,\pi) = \sum_{t=9}^{12} c_j^t \cdot x_j^t + \sum_{t=9}^{14} \sum_{s=\max(9,t-2)}^{\min(t,12)} x_j^s \cdot \pi^t =$$

$$c_j^9 x_j^9 + c_j^{10} x_j^{10} + c_j^{11} x_j^{11} + c_j^{12} x_j^{12} +$$
$$\pi^9 x_j^9 + \pi^{10}\left(x_j^9 + x_j^{10}\right) + \pi^{11}\left(x_j^9 + x_j^{10} + x_j^{11}\right) +$$
$$\pi^{12}\left(x_j^{10} + x_j^{11} + x_j^{12}\right) + \pi^{13}\left(x_j^{11} + x_j^{12}\right) + \pi^{14}\left(x_j^{12}\right) = \quad (4.6)$$
$$x_j^9\left(c_j^9 + \pi^9 + \pi^{10} + \pi^{11}\right) +$$
$$x_j^{10}\left(c_j^{10} + \pi^{10} + \pi^{11} + \pi^{12}\right) +$$
$$x_j^{11}\left(c_j^{11} + \pi^{11} + \pi^{12} + \pi^{13}\right) +$$
$$x_j^{12}(c_j^{12} + \pi^{12} + \pi^{13} + \pi^{14})$$

Para resolver o subproblema acima exemplificado, basta verificar em (4.6) o valor da função objetivo para todas as possíveis datas de início da tarefa *j*, verificando aquela que minimiza a função $D(j,\pi)$. Neste caso as possíveis datas de início são 9, 10, 11 ou 12. Ao se escolher uma data genérica t, os elementos π^t, π^{t+1}, ..., π^{t+p_j-1} serão computados na função objetivo juntamente com c_j^t. A soma dos multiplicadores de Lagrange indica a penalização total decorrente das violações de capacidade nas datas $t, t+1, \ldots, t+p_j-1$. Ao se programar uma tarefa, a data que totalizar a menor violação de capacidade conjuntamente com a penalização associada ao nível de serviço será escolhida.

4.3.4 Resolução do Problema Dual

Para resolver o problema dual, no espaço de solução das variáveis π, será aplicado o método do subgradiente, o qual atualizará o valor de π, de acordo com a expressão (4.7).

$$\pi^{n+1} = [\pi^n + \alpha^n g^n(\pi^n)] \tag{4.7}$$

Em que:

- n – número da iteração
- g^n – subgradiente de D para um dado π, cujo *t*-ésimo componente é dado por:

$$\left(\sum_{\substack{j=1: \\ t \geq r_j}}^{n} \sum_{s=\max(r_j, t-p_j+1)}^{\min(t, T-p_j+1)} x_j^s \right) - m \tag{4.8}$$

- α^n – é o passo na iteração *n*, dado por:

$$\alpha^n = \lambda \frac{\bar{L} - L^n}{g^n(\pi^n)^T g^n(\pi^n)} \quad 0 < \lambda < 2 \tag{4.9}$$

- $\bar{L}$ – é o limitante superior da função objetivo do problema original, calculado pela solução heurística no início do algoritmo.
- L^n – é o limitante inferior da função objetivo do problema original, na iteração n, cujo valor é dado por:

$$min\left(-\sum_{t=1}^{T} \pi^t m + \sum_{j=1}^{n} D(j, \pi) \right) \tag{4.10}$$

- λ – é um fator de ajuste do tamanho do passo, que será reduzido pela metade cada vez que a iteração não melhorar o limitante inferior por um número consecutivo de vezes. Esse número é um parâmetro que influi no critério de parada e, portanto deverá ser calibrado apropriadamente. Pode ser interessante, em termos de desempenho do algoritmo, que esse número varie de acordo com o tamanho do passo ou outro fator relevante.
- Por último, o operador $[\phi]^+$ irá retornar $\max(0, \phi)$.

4.4 Geração de Soluções Viáveis & Busca Local

O método de solução desenvolvido nesta pesquisa tem como proposta explorar a vizinhança das soluções duais, geradas pela heurística lagrangiana, visando encontrar soluções de boa qualidade no espaço de solução do problema original. Tendo em vista que grande parte das soluções duais é inviável, foi proposto um algoritmo que transforma a solução inviável em viável e, em seguida, a busca local é realizada.

No caso de uma solução viável estar localizada em uma região distante da solução ótima, o algoritmo de busca poderá gastar muito tempo de processamento para encontrar regiões mais promissoras. Em função disso, foi proposto um critério mediante o qual a solução dual será ou não convertida em solução primal viável e, posteriormente, submetida à busca local. Este consiste comparar o valor da função objetivo da solução dual que acabou de ser gerada com o valor da função objetivo da solução ótima estimada até o momento (máximo valor da função objetivo de todas as soluções duais geradas até o momento). Caso valor da função objetivo da solução dual, dividido pelo valor da função objetivo do limitante inferior seja maior do que α, a busca local é executada. Onde o parâmetro α, um valor que varia de 0 a 1, pode ser calibrado de forma a evitar grande esforço computacional com soluções duais muito distantes da região viável.

Nas próximas subseções as heurísticas serão explicadas em detalhe.

4.4.1 Procedimento para Geração de Soluções Viáveis

O algoritmo usado na geração de soluções viáveis foi criado pensando na estrutura do problema relaxado e nas características das soluções obtidas.

O problema original, resolvido pelo método da relaxação lagrangiana, tinha apenas um tipo de restrição violada, a saber, a restrição de capacidade. Assim sendo, as soluções geradas pela resolução do problema relaxado atendiam a restrição de atendimento integral da demanda sem necessariamente respeitar a capacidade das embarcações ao longo do horizonte de planejamento.

A heurística que transforma uma possível solução inviável em viável identifica os instantes em que a restrição de capacidade é violada, e posterga o início de uma ou mais tarefas realizadas nestes instantes. O deslocamento das tarefas é feito percorrendo a linha do tempo até que em nenhum momento a restrição de capacidade do problema original fosse violada.

Os critérios utilizados para definir quais tarefas terão seu início postergado são dois. O primeiro é a data de início da tarefa sugerido na resolução do problema relaxado, e o segundo é baseado no custo associado a postergar o início da tarefa em um dia. Caso as tarefas tenham a mesma data de início sugerida, a tarefa que manterá o início da execução será aquela que tiver a maior penalidade associada a atrasar em um dia o seu início.

É importante observar que as soluções do problema relaxado podem ser inviáveis ao não respeitar a restrição de capacidade das embarcações. Por isso, não é necessário verificar se as tarefas estão sendo iniciadas antes da data de liberação das mesmas, fazendo com que este parâmetro de liberação não seja levado em conta no algoritmo que gera soluções viáveis a partir das soluções do problema relaxado.

Para facilitar o entendimento dos critérios utilizados para gerar soluções viáveis, conforme descrito acima, será proposto um exemplo, com 5 tarefas e 2 embarcações, onde uma solução inviável foi proposto e onde aplicaremos o algoritmo que a torna viável. As informações que definem as características deste problema exemplo estão indicadas na Tabela 4.1 e as passagens do algoritmo utilizado estão representadas na Figura 4.1.

No exemplo proposto tem-se 5 tarefas programadas e apenas 2 embarcações. De acordo com o que pode ser visto na fase 1 da Figura 4.1, há violação da capacidade das embarcações já no instante 1, quando estão previstas a execução das tarefas A, C e D. Para definir quais tarefas permanecerão programadas, de acordo com a programação original, são verificados os critérios "início sugerido" e "penalidade" associada a postergar o início da tarefa em 1 dia. Dessa forma, prevalecem as tarefas C e D, as quais serão executadas no instante 1, respectivamente, pelas embarcações β e α.

Tarefa	Duração	Data de Entrega	Penalidade	Liberação	Início Sugerido	Final da Tarefa
A	6	6	1	0	1	6
B	4	6	4	2	7	10
C	9	9	5	0	1	9
D	7	7	9	0	1	7
E	5	8	8	3	7	11

Tabela 4.1 - Problema Exemplo: 5 Tarefas e 2 Embarcações

Com as tarefas C e D sendo iniciadas no instante 1, o instante mais cedo em que haverá uma embarcação disponível para executar novas tarefas é o instante 8. Observando a programação original, constata-se que é necessário postergar o início das tarefas A, B e E pelo menos até o instante 8, quando o critério para escolha da próxima tarefa será aplicado novamente. Neste instante, tem-se uma embarcação disponível (α) e 3 tarefas candidatas (A, B e E). Como a tarefa A tem a menor data de início de acordo com a programação original, ela será alocada neste instante,

fazendo com que as demais tarefas sejam postergadas até o próximo instante em que haverá uma embarcação disponível.

No instante 10 a embarcação β estará liberada, e haverá duas tarefas em fila (B e E). Como as duas tarefas possuem o mesmo início previsto, o critério que definiu pela alocação da tarefa E foi baseado no fato desta tarefa ter uma maior penalidade. Por fim, a tarefa B é alocada no instante 14, quando a embarcação α ficou disponível. Dessa forma, tem-se uma solução viável para o problema de programação de tarefas.

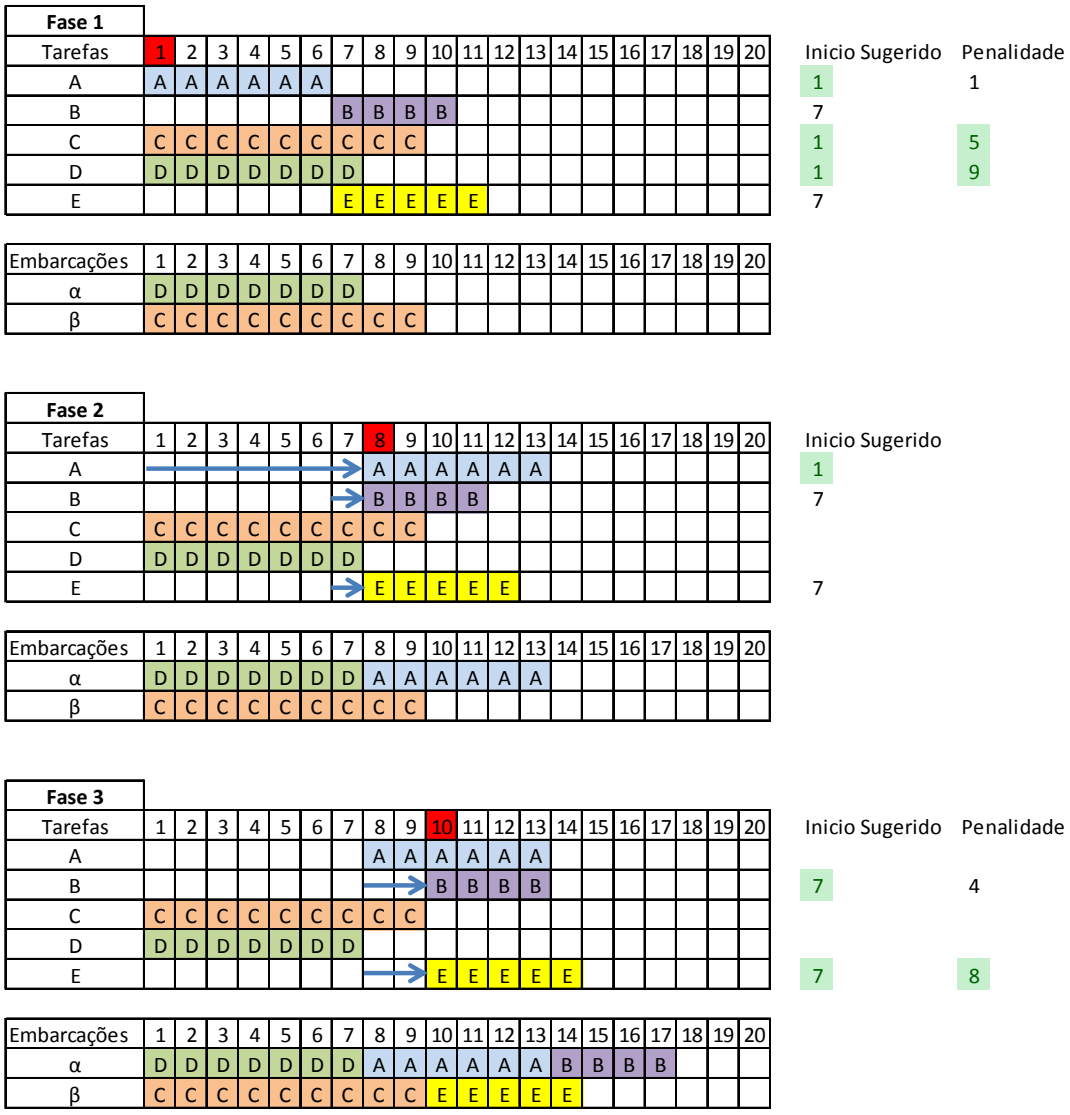

Figura 4.1 - Etapas para geração de solução viável para exemplo proposto

4.4.2 Procedimentos de Busca Local

A busca local implementada consistiu em uma adaptação do "variable neighborhood search" (VNS). O VNS é uma meta-heurística que se baseia na exploração de diferentes estruturas de vizinhanças a partir de uma solução inicial, visando encontrar soluções de melhor qualidade até atingir o ponto ótimo (Hansen, 2003).

Nessa pesquisa optou-se por estruturar o VNS a partir da proposta de Hansen (2003) indicada na Figura 4.2. Essa estrutura se diferencia do VNS básico por realizar mais de uma busca local para cada nova vizinhança analisada.

Inicialização. Selecionar um conjunto de estruturas de vizinhança $N_k, k = 1, \dots, k_{max}$, que serão usadas na fase de agitar, e selecionar o conjunto de vizinhanças $N_\ell, \ell = 1, \dots, \ell_{max}$, que serão usadas na fase de busca local; Informar uma solução inicial *x*, definir um critério de parada.

Até que o critério de parada seja atendido, ***Repetir:***

(1) Fazer $k \leftarrow 1$;

(2) Até que $k = k_{max}$, repetir os passos:

(a) *Agitar*. Gerar um ponto x' aleatoriamente na *k*-ésima vizinhança de x ($x' \in N_k(x)$)

(b) *Busca Local*. Aplicar algum método de busca local tendo x' como solução inicial; a melhor solução na vizinhança de x' será designada de x''

(b1) Fazer $\ell \leftarrow 1$

(b2) Até que $l = l_{max}$, repetir os passos

- Explorar a vizinhança e encontrar a melhor solução possível x'' da solução x' na vizinhança $N_l(x')$.

- Mover ou não. Se $f(x'') < f(x')$ então $x' \leftarrow x''$ e $\ell \leftarrow 1$; caso contrário $\ell \leftarrow \ell + 1$

(c) *Mover ou não*. Se este ótimo local for melhor que a melhor solução já obtida, mover para este local ($x \leftarrow x'$) e continuar a busca com $N_1(k \leftarrow 1)$; caso contrário, fazer $k \leftarrow k + 1$;

Figura 4.2 – Estrutura Geral do VNS (adaptado de Hansen, 2003)

A adaptação feita em relação ao método VNS indicado na Figura 4.2 se dá no momento da geração de soluções na vizinhança (“agitar”). No algoritmo desenvolvido nesta pesquisa, a solução inicial é advinda da heurística lagrangiana, a qual é transformada em viável, ao passar pelos critérios de aceitação mencionados.

O VNS empregado nesta pesquisa utiliza diferentes estruturas de vizinhança, o que não restringe a busca a uma região específica da vizinhança que se pretende explorar. Quanto maior for o número de soluções que se consiga gerar e avaliar, a partir das diferentes estruturas de vizinhanças, menor é a probabilidade do algoritmo ficar preso a um possível ótimo local encontrado durante a execução dos procedimentos de busca.

Nesta pesquisa optou-se pela construção de três estruturas de vizinhança distintas. A primeira delas se refere ao reposicionamento de uma única tarefa, determinada por sorteio, e alocada na posição que minimiza a função objetivo. A segunda estrutura de busca, mais complexa do ponto de vista computacional, baseia-se no reposicionamento simultâneo de duas tarefas sorteadas aleatoriamente, de forma a melhorar a função objetivo. A última busca compara a troca de posição entre todos os pares possíveis de tarefas avaliando qual a melhor troca.

Para aumentar a eficiência do ponto de vista computacional, é necessário iniciar pelas buscas mais simples, a fim de encontrar soluções de boa qualidade sem grande necessidade de processamento. A partir do momento em que as buscas simples não surtam mais efeito, parte-se

para o uso das buscas mais complexas que verificam a qualidade de soluções mais afastadas da solução que está sendo examinada.

A Figura 4.3 mostra de forma esquemática o espaço de soluções viáveis e a abrangência das estruturas de vizinhança das diferentes buscas locais. Em seguida, serão melhor detalhadas as estruturas utilizadas.

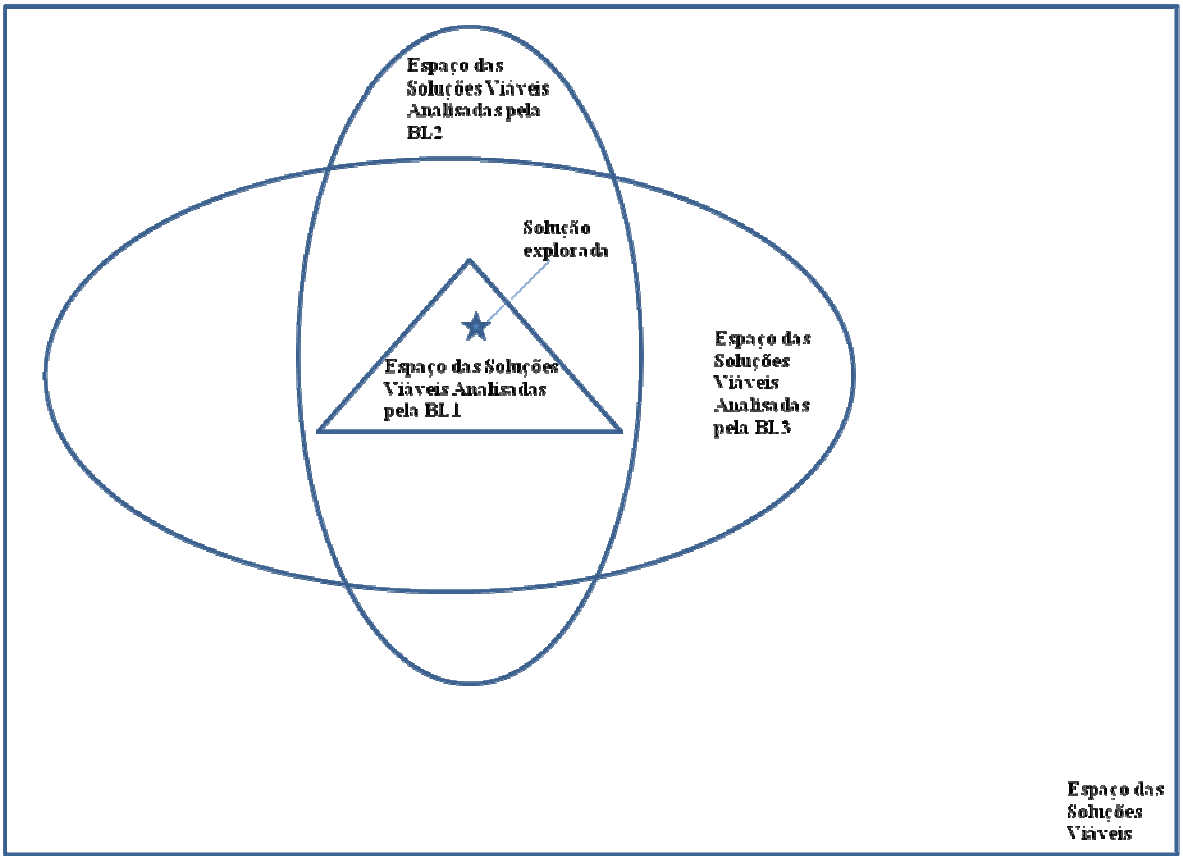

Figura 4.3 - Espaço das Soluções Viáveis e Buscas Locais

4.4.2.1 Busca Local 1 (BL1) – Reposicionamento de Uma Tarefa

A primeira estrutura de vizinhança, a busca local 1 (BL1), escolhe uma tarefa qualquer do problema, de forma aleatória, e retira essa tarefa da embarcação em que ela está alocada. Em seguida inicia-se a rotina de colocar a tarefa em todas as posições possíveis de todas as embarcações disponíveis, a fim de determinar a configuração que implica em menor valor da função objetivo. Caso o custo dessa configuração seja menor do

que a solução que está sendo analisada, é considerado que a busca local teve sucesso.

Como a estrutura desse método conta com um elemento aleatório (sorteio da tarefa que será realocada), a execução dessa busca local uma única vez não é suficiente para concluir que o insucesso de uma busca implique em sucessivos insucessos, mesmo que se esteja partindo de uma mesma solução a ser analisada. O número de vezes que a BL1 é executada antes de partir para uma nova estratégia de busca local é um parâmetro de entrada do algoritmo, e pode variar de acordo com as características do problema que está sendo resolvido como, por exemplo, o número de tarefas e o número de embarcações.

A Figura 4.4 exemplifica a forma como a BL1 é executada. Neste caso sorteou-se a tarefa H, aleatoriamente e, em seguida, essa tarefa foi alocada na melhor posição possível. Caso a tarefa tivesse sido devolvida à posição original, a busca teria tido um insucesso.

Toda vez que a busca local de reposicionamento de uma tarefa é executada, o esforço computacional envolvido no processo depende basicamente do número de tarefas do problema, uma vez que a tarefa sorteada será testada em n posições, sendo n o número de tarefas do problema.

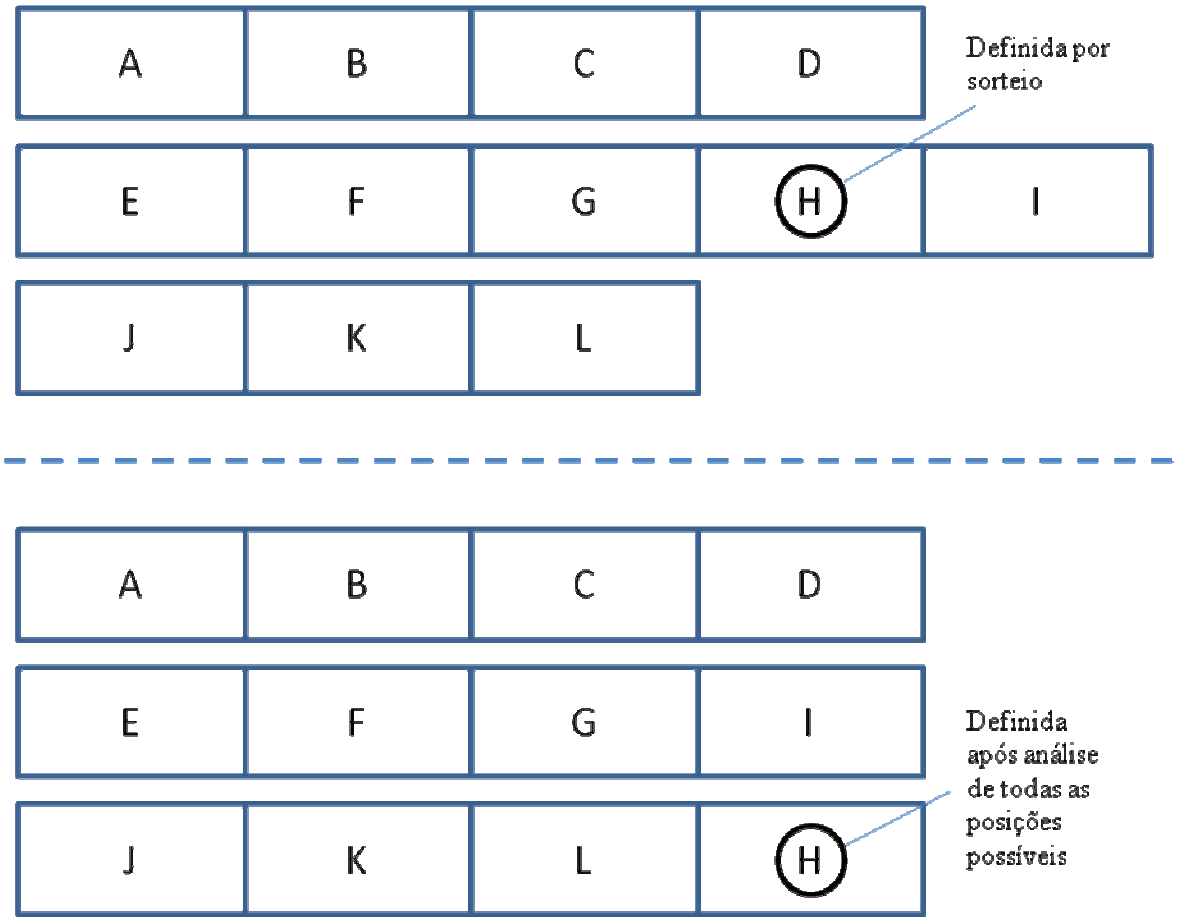

Figura 4.4 - Busca Local 1: Reposicionamento de tarefa

4.4.2.2 Busca Local 2 (BL2) – Recolocação de duas tarefas

A segunda estrutura de vizinhança proposta, a busca local 2 (BL2), seleciona duas tarefas de maneira aleatória, observando a condição de que estas tarefas devam estar alocadas, originalmente, em embarcações diferentes. As duas tarefas sorteadas são retiradas das embarcações em que se encontravam, gerando uma configuração denominada de alocação inicial (a menos das 2 tarefas excluídas), como pode ser vista na Figura 4.5.

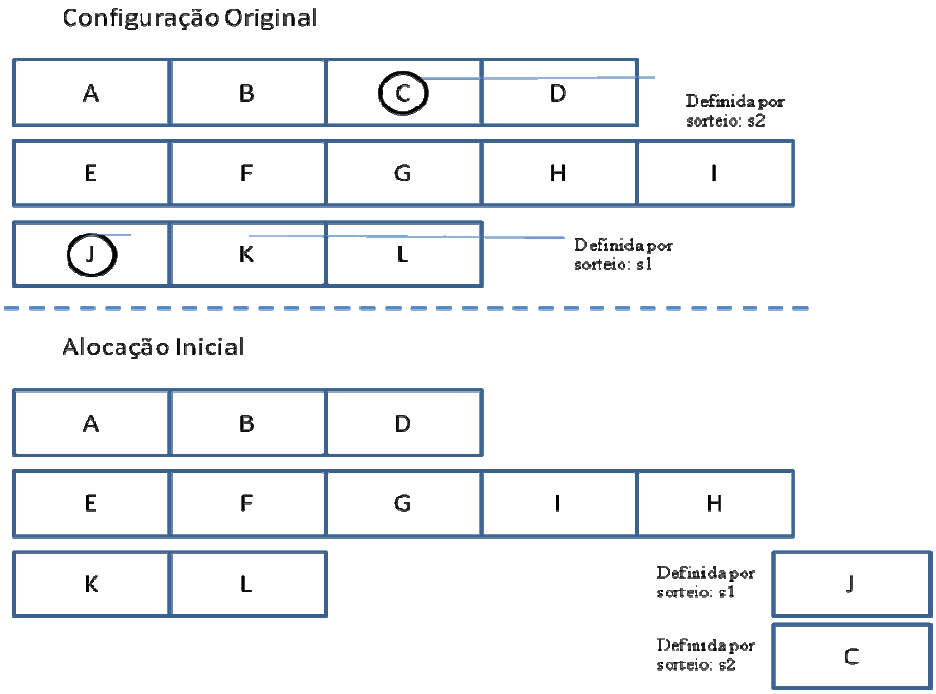

Figura 4.5 - Busca Local 2: Alocação Inicial

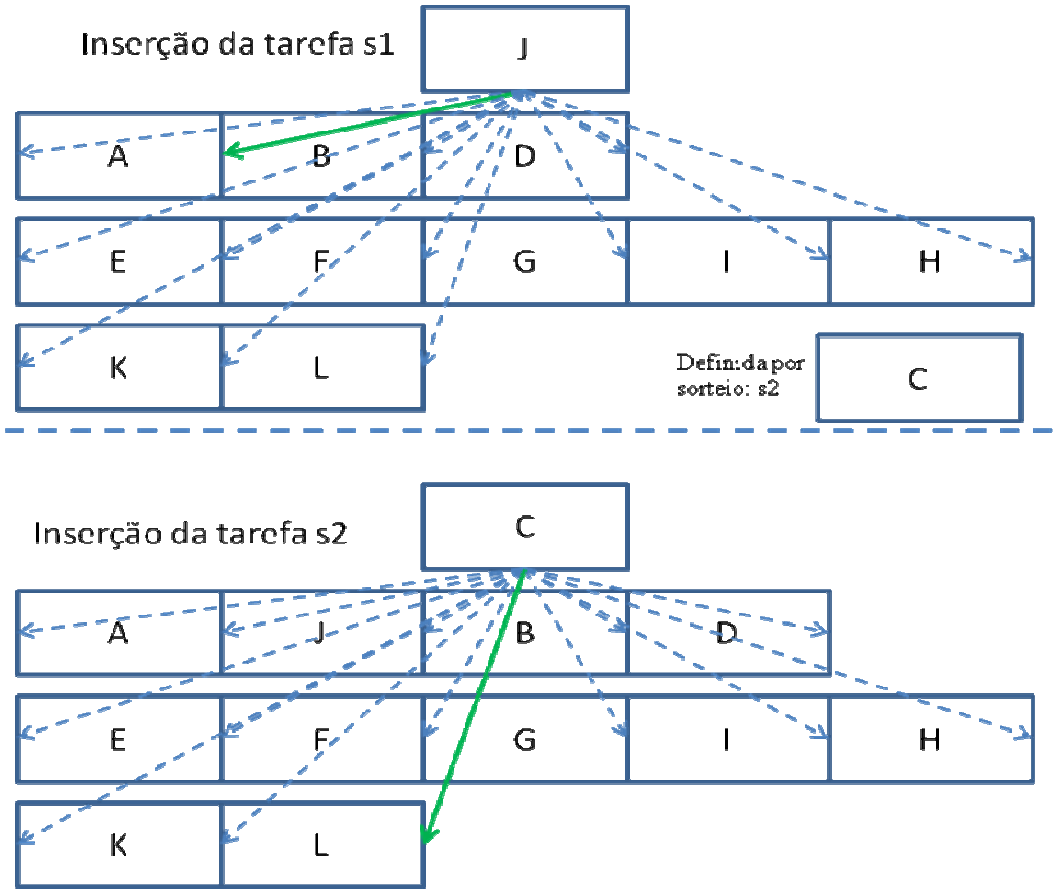

Figura 4.6 - Busca Local 2: Inserções de s1 e s2

Para definir qual a melhor forma de alocar as tarefas, parte-se da alocação inicial procurando qual a melhor posição para recolocar a primeira tarefa que foi sorteada (s1). Definida a posição de inserção da tarefa s1 que minimiza a função objetivo, faz-se a inserção da mesma. Em seguida, verifica-se qual a melhor posição para inserir a segunda tarefa sorteada (s2). Ao encontrar a melhor posição para inserir a tarefa s2, obtem-se uma solução viável para o problema. A Figura 4.6 fornece uma visão esquemática das inserções acima citadas. Caso a solução viável encontrada forneça uma função objetivo menor que a melhor função objetivo encontrada até o momento, atualiza-se a melhor solução viável com a solução encontrada e considera-se que a busca local teve sucesso.

A Busca Local de recolocação de duas tarefas, contudo, ainda não é considerada concluída. Partindo-se da alocação original, inicia-se agora a alocação das duas tarefas sorteadas pela tarefa s2 para, depois de definida a sua melhor posição de inserção avaliar a melhor posição para inserção da tarefa s1. Obtém-se, ao final dessas duas alocações otimizadas, uma solução viável que, caso tenha o valor da função objetivo menor que a solução corrente, caracteriza a busca local como um sucesso, atualizando a melhor solução encontrada até o momento.

É importante verificar a solução obtida com a inserção das tarefas começando também pela tarefa s2, pois estamos explorando uma região mais abrangente do espaço de soluções viáveis e, consequentemente, aumentando a chance de encontrar melhorias para a solução pesquisada.

A estrutura da BL2, assim como a busca local caracterizada no item anterior, depende de elementos sorteados de forma aleatória, fazendo com que a execução do método mais de uma vez possa gerar diferentes resultados. Neste caso, para considerar que a busca local foi incapaz de ter sucesso na busca por uma solução viável de melhor qualidade, é necessário repetir a busca algumas vezes. O número de repetições que,

idealmente, devem ser realizadas, pode variar de acordo com o porte do problema sendo necessária a calibragem desses parâmetros para melhorar os resultados e o desempenho do algoritmo.

Quando se executa a busca local de reposicionamento de duas tarefas, o esforço computacional envolvido é aproximadamente 4 vezes maior que a busca local envolvendo o reposicionamento de uma só tarefa. No caso da BL2 são testadas n–1 posições para inserção da primeira tarefa e n posições para inserção da segunda. O procedimento é repetido iniciando-se pela segunda tarefa sorteada, para depois inserir a primeira tarefa, implicando em 4n – 2 comparações.

4.4.2.3 Inversão de Duas Tarefas

A última estrutura de busca local implementada, a busca local 3 (BL3), embora de fácil compreensão, implica em grande esforço computacional. Nesta busca, analisa-se o valor da função objetivo para todas as trocas de pares de tarefas, independentemente de estarem na mesma embarcação ou não, ou seja, inverte-se a posição do par de tarefas escolhido.

A quantidade de comparações a serem feitas depende exclusivamente do número de tarefas e pode ser determinada pelo total de combinações de tarefas 2 a 2 dado por n*(n-1). Vale a pena observar que, para as maiores instâncias que se pretende resolver (50 tarefas e 10 embarcações), a título de esforço computacional, o número de comparações a ser feitas cada vez que essa busca local é executada é de 2450 contra 198 da segunda busca local e 50 da primeira busca local. Isto posto, é necessário analisar as vantagens e desvantagens da utilização das buscas locais dependendo do porte do problema que se pretende resolver.

Esse tipo de busca não depende de valores aleatórios de forma que para uma configuração de solução viável qualquer, o resultado obtido com algoritmo é sempre o mesmo, não sendo necessário repetir a busca local a não ser que tenhamos modificações na solução que está sendo analisada.

4.4.2.4 Calibragem de Parâmetros

Como explicado anteriormente, nem toda solução proposta no método da relaxação lagrangiana é transformada em uma solução viável. Para determinar se uma dada solução do problema relaxado será transformada em solução viável e, futuramente, será submetida à busca local desenvolvida, é necessário que essa solução esteja próxima do valor do limitante inferior determinado até o momento.

Um dos parâmetros que devem ser calibrados para bom funcionamento do algoritmo diz respeito à quão próxima uma solução gerada na resolução do problema relaxado deve estar do limitante inferior. Para estabelecer esse parâmetro é necessário ter em mente que o valor do parâmetro afeta a quantidade de vizinhanças geradas para o uso do algoritmo de busca local, impactando no tempo de processamento, e na qualidade das soluções propostas. Ao estabelecer que uma solução candidata deva estar muito próxima do limitante inferior calculado até o momento, espera-se conseguir ganho em tempo de processamento, mas por outro lado, pode acontecer de uma vizinhança que daria origem a uma solução que diminuiria o limitante superior ser ignorada. Por outro lado, se a busca local for conduzida em toda solução encontrada, estaremos impedindo que uma vizinhança promissora seja ignorada, mas haverá grande impacto no tempo de processamento, uma vez que em várias iterações a busca estará sendo conduzida em soluções de baixa qualidade.

Outros parâmetros importantes dizem respeito ao critério de parada utilizado na busca local. Esses parâmetros indicam quantas vezes cada estrutura de busca local deverá ser realizada antes de considerar que a busca, como um todo, para a solução que está sendo avaliada, não teve sucesso. É importante destacar que a estrutura de Busca Local 3 só precisará ser realizada uma única vez para uma dada solução, já que o resultado obtido com a mesma será sempre o mesmo, devido à inexistência de fatores randômicos.

Os parâmetros acima explicados serão calibrados de forma empírica por meio de testes, variando a quantidade de tarefas e embarcações disponíveis e observando tanto os tempos de processamento, como a qualidade das soluções obtidas.

4.5 **Resumo do Procedimento**

Para facilitar a compreensão da metodologia apresentada, foi proposto o fluxograma mostrado na Figura 4.7. Pode ser visto, neste fluxograma, como estão relacionadas a heurística construtiva, a relaxação lagrangiana, o método de busca local, além de parâmetros como o multiplicador do passo (utilizado na relaxação lagrangiana), o contador de iterações, usado como um dos critérios de parada e a atualização da melhor solução viável e da estimativa da solução viável para o problema.

De forma resumida, a metodologia proposta para solução do problema funciona assim:

A. Inicialização dos parâmetros do modelo.
B. Geração de solução viável com uso da heurística construtiva.
C. Verificação dos critérios de parada.
D. Estimativa da solução ótima com uso da relaxação lagrangiana.

E. Análise da solução relaxada: caso a solução tenha bom potencial, vai para F, caso contrário vai para C.
F. Gera solução viável a partir da solução relaxada.
G. Uso da heurística de busca local para tentar melhor a solução viável gerada. Volta para C

Na etapa A, de inicialização dos parâmetros do modelo, são definidos alguns parâmetros utilizados no algoritmo de solução. É importante ressaltar que como se inicia a resolução do problema sem o conhecimento de uma solução viável, e sem a certeza de existir uma solução viável para o horizonte de planejamento proposto, considera-se que o valor da função objetivo da melhor solução viável encontrada até o momento vale infinito. Caso algum problema esteja sendo resolvido para o qual uma solução viável seja conhecida, é possível inseri-la na inicialização do algoritmo.

Na etapa B uma solução é gerada por meio de uma heurística gulosa, que visa gerar uma solução inicial de boa qualidade para o problema. A solução proposta pode ser inviável por não respeitar o horizonte de planejamento. Neste caso uma nova programação é gerada, desta vez focando na redução do tempo necessário para que todas as tarefas sejam executadas. Por fim, é aplicada uma heurística de busca local para melhorar a qualidade do limitante superior que alimentará a relaxação lagrangiana.

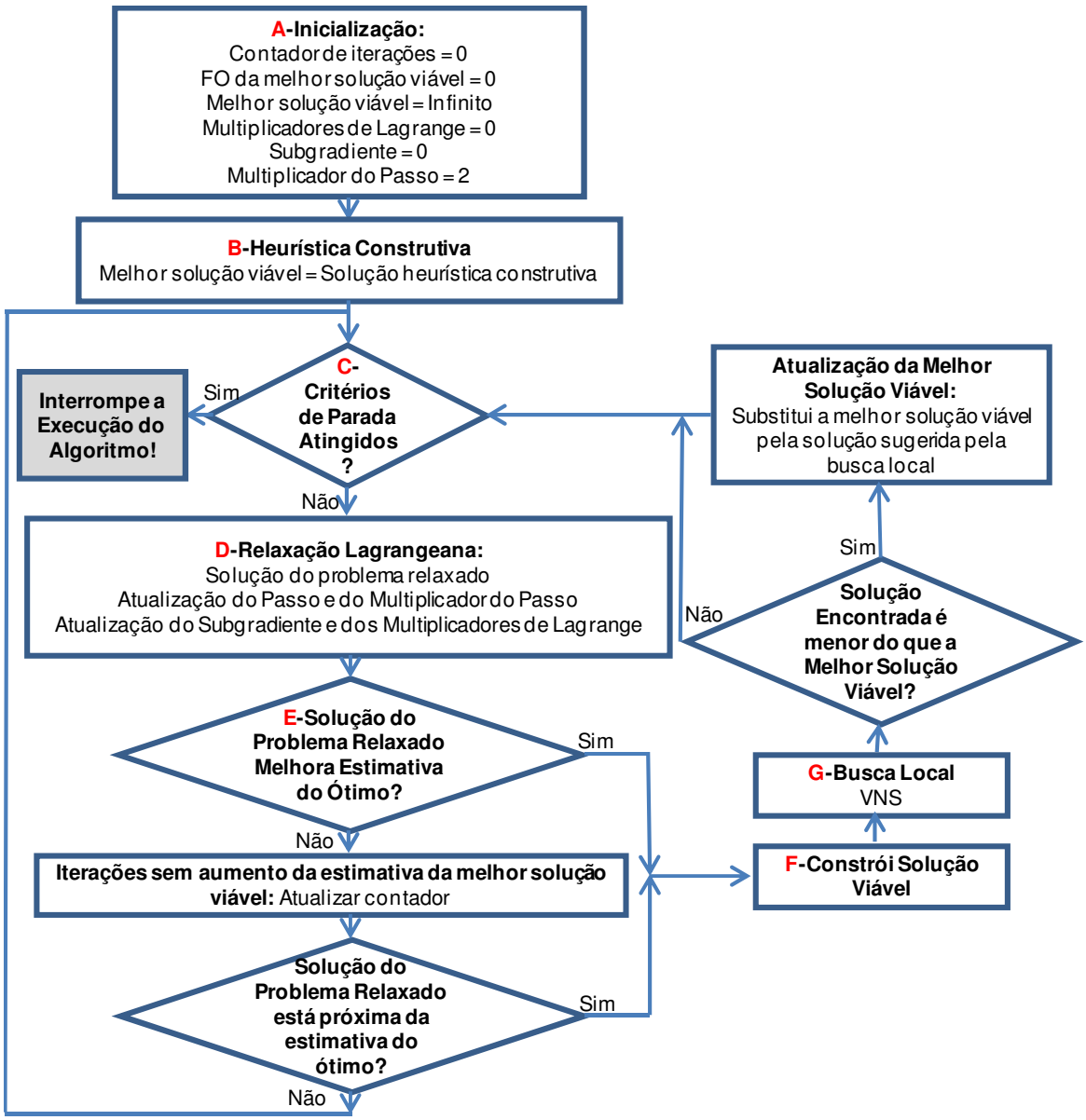

Figura 4.7 - Fluxograma da Metodologia

Em seguida, na etapa C, é feita a verificação dos dois critérios de parada. Caso o tamanho do passo utilizado na relaxação lagrangiana esteja menor do que o passo mínimo estabelecido na calibragem, ou o limitante inferior esteja igual ao limitante superior, a execução do algoritmo é interrompida. Em caso contrário, a etapa D é chamada, quando a relaxação lagrangiana é aplicada novamente: atualiza-se o tamanho do passo, atualizam-se os multiplicadores de Lagrange e resolve-se o problema relaxado por enumeração exaustiva, definindo o melhor instante para início de cada uma das tarefas.

A solução gerada pela relaxação lagrangiana é avaliada. Caso o valor da função objetivo do problema relaxado, que acabou de ser resolvido, melhore o limitante inferior (critério utilizado na etapa D), parte-se para a etapa F, em que uma nova solução viável será gerada. Caso o valor da função objetivo do problema relaxado que acabou de ser resolvido esteja próxima do melhor limitante calculado até o momento (o conceito de próximo é definido por um parâmetro α, detalhado na seção 4.4), é gerada uma nova solução viável na etapa F. Caso contrário, nenhuma solução viável será gerada a partir desta solução relaxada e encerra-se um ciclo voltando para a etapa C.

Como descrito acima, nos casos em que a solução relaxada for considerada de boa qualidade, na etapa F, uma heurística é aplicada sobre essas soluções fazendo com que as mesmas se tornem viáveis, para em seguida, iniciar a etapa G.

A etapa G utiliza-se da heurística de busca local para tentar melhorar a solução viável gerada e, em seguida, um ciclo é encerrado, retornando para a etapa C, quando os critérios de parada serão novamente verificados.

5 APLICAÇÃO DOS MÉTODOS DE SOLUÇÃO

Neste capítulo serão apresentados os resultados obtidos com o método de solução proposto na seção anterior para resolução do problema de PROGRAMAÇÃO DE FROTA DE EMBARCAÇÕES DE LANÇAMENTO DE DUTOS. Inicialmente, será apresentada a estrutura dos cenários utilizados no teste do método de solução, assim como os parâmetros utilizados para geração desses cenários. Em seguida serão apresentados os resultados alcançados pelo algoritmo para os respectivos cenários. Por fim serão discutidos os resultados obtidos.

Para avaliar o desempenho do método de solução serão feitos testes variando-se o número de tarefas, a quantidade de embarcações disponíveis para realização dessas tarefas e um fator de folga, que implica no número de dias ou datas que uma tarefa pode ter seu início programado sem que haja penalização na função objetivo. O horizonte de planejamento (hp) de todas as instâncias de teste foi fixado em 120 dias.

A metodologia utilizada para geração das instâncias de teste foi adaptada do trabalho de Jouglet e Savourey (2011). O número de tarefas (n) variou da seguinte forma: n = 20, 30, 50 e 100. Além de variar o número de tarefas nos problemas, variou-se a quantidade de embarcações (m): m = 4, 5, 6, 8, 10.

Os parâmetros associados às tarefas, em cada cenário, foram gerados de forma aleatória, em função dos fatores α (parâmetro de liberação de tarefas no instante 0, indica a probabilidade das tarefas estarem liberadas no início do horizonte de planejamento), β (parâmetro de folga, que afeta a quantidade de datas disponíveis para início da tarefa sem incorrer em penalidade), da seguinte forma:

- Duração máxima (Dm): Este parâmetro é calculado em função do número de embarcações, do número de tarefas e do horizonte de planejamento, e é utilizado para estimar a

máxima duração das tarefas de forma a evitar a geração de instâncias inviáveis. Adotou-se a seguinte fórmula:

$$Dm = \frac{hp \times \propto^2 \times m}{n}$$

- Duração das tarefas (p_j): número aleatório inteiro no intervalo que varia de 1 à duração máxima (Dm) da instância.
- Penalidade por atraso na finalização da tarefa (w_j): Distribuição uniforme variando de 1 a 10.
- Data de liberação das tarefas (r_j) é calculado, tarefa a tarefa, da seguinte forma: o parâmetro α define a probabilidade da tarefa ser liberada no instante 0. Caso a tarefa não seja liberada no instante 0 ela será liberada em um instante definido por uma número aleatório inteiro entre os datas 0 e $hp - p_j$.
- Data de entrega das tarefas (Dj): Soma da data de liberação (r_j), com a duração da tarefa (p_j), com um número aleatório inteiro entre 0 e a duração da tarefa (p_j) multiplicado pelo fator de folga β, ou seja: $D_j = r_j + p_j + aleat(0; p_j \times \beta)$.

Para avaliar o desempenho do algoritmo proposto, foram utilizados seis indicadores:

- GAP: Representa a diferença entre o limitante superior e o limitante inferior, sendo calculado como: $GAP = \frac{Limitante\ Superior - Limitante\ Inferior}{Limitante\ Superior}$.
- GAP^+: Representa a diferença entre o limitante superior e o valor da função objetivo da solução ótima do problema (determinada por meio de um pacote de otimização). Foi definido da seguinte forma: $GAP^+ = \frac{Limitante\ Superior - F.O.Otimo}{Limitante\ Superior}$
- GAP^-: Representa a diferença entre o limitante inferior e o valor da função objetivo da solução ótima do problema

(determinada por meio de um pacote de otimização). Foi definido como: $GAP^{-} = \frac{F.O.Otimo - Limitante\ Inferior}{F.O.Otimo}$

- GAP^{0}: Representa a diferença entre o valor da função objetivo da solução obtida com a heurística construtiva e o valor da função objetivo da solução ótima do problema (determinada por meio de um pacote de otimização). Foi definido como: $GAP^{0} = \frac{F.O.Solução\ Inicial - F.O.Otimo}{F.O.Otimo}$
- Número de Iterações: Representa a quantidade de vezes que o problema lagrangiano foi resolvido ao longo do procedimento.
- Tempo de Processamento

A quantidade de instâncias testadas, variando os parâmetros supracitados, foi de 240. Os resultados obtidos não serão apresentados individualmente, mas agrupados de acordo com os parâmetros que definiram a estrutura de cada um. Ainda neste capítulo, serão discutidos os resultados apresentados.

5.1 **Resultados dos Testes Computacionais**

O algoritmo proposto neste trabalho foi desenvolvido em linguagem de programação VBA (Excel). O hardware utilizado para processar as instâncias de teste foi um processador Intel® Core™ i7 2.80GHz e 16.374MB de memória RAM DDR3 SDRAM.

Conforme descrito na introdução deste capítulo, foram geradas diversas instâncias de teste. Será chamado de cenário, o conjunto de dez instâncias, geradas aleatoriamente, de acordo com a combinação de quatro parâmetros que definem as características do problema: número de tarefas (n), número de embarcações (m), e parâmetros de dispersão α e β. Os valores dos parâmetros α e β utilizados na geração das instâncias

de teste foram definidos de forma a gerar problemas de diferentes portes e características, mantendo aspectos do problema real, como por exemplo, o fato da maior parte das tarefas já estar disponível no início do horizonte de planejamento.

As combinações dos parâmetros utilizados para gerar um cenário, assim como os indicadores de cada cenário podem ser vistos abaixo. Na Tabela 5.1 está o resultado médio, na Tabela 5.2 estão os piores casos (Gap's Máximos) e na Tabela 5.3 estão os melhores casos (Gap's Mínimos) de cada cenário.

Tabela 5.1- Resultado Médio dos Cenários

Cenário	Tarefas (n)	Embarcações (m)	Fator de Folga (α)	Fator de Liberação (β)	GAP Médio	GAP + Médio	GAP - Médio	GAP 0 Médio	Número de Iterações Médio	Tempo de Processamento Médio (s)
1	20	4	0%	90%	0,02%	0,01%	0,01%	2,55%	339	11
2	20	4	50%	90%	0,05%	0,01%	0,04%	6,66%	383	13
3	20	4	100%	90%	0,49%	0,22%	0,27%	7,04%	610	19
4	20	5	0%	90%	0,01%	0,00%	0,01%	2,82%	361	12
5	20	5	50%	90%	0,11%	0,02%	0,09%	6,78%	603	18
6	20	5	100%	90%	0,30%	0,02%	0,28%	16,01%	567	17
7	30	5	0%	90%	0,07%	0,07%	0,01%	3,50%	666	41
8	30	5	50%	90%	0,01%	0,01%	0,00%	2,80%	588	31
9	30	5	100%	90%	0,05%	0,01%	0,04%	4,31%	509	30
10	30	6	0%	90%	0,01%	0,00%	0,00%	2,87%	513	27
11	30	6	50%	90%	0,00%	0,00%	0,00%	5,58%	449	24
12	30	6	100%	90%	0,61%	0,30%	0,31%	10,04%	768	44
13	50	6	0%	90%	0,02%	0,02%	0,00%	2,21%	765	797
14	50	6	50%	90%	0,05%	0,04%	0,01%	4,24%	804	895
15	50	6	100%	90%	0,09%	0,07%	0,03%	4,21%	1.008	1.064
16	50	8	0%	90%	0,02%	0,02%	0,00%	3,16%	763	771
17	50	8	50%	90%	0,06%	0,05%	0,01%	3,95%	769	775
18	50	8	100%	90%	0,17%	0,14%	0,03%	7,74%	901	890
19	100	8	0%	90%	0,06%	0,06%	0,00%	1,82%	1.073	597
20	100	8	50%	90%	0,06%	0,06%	0,00%	2,36%	1.078	920
21	100	8	100%	90%	0,06%	0,06%	0,00%	1,75%	1.061	3.025
22	100	10	0%	90%	0,06%	0,06%	0,00%	2,10%	1.090	2.980
23	100	10	50%	90%	0,06%	0,06%	0,00%	2,29%	1.072	2.823
24	100	10	100%	90%	0,12%	0,12%	0,01%	3,26%	1.054	2.905

A Tabela 5.4 apresenta os resultados mensurados com os indicadores propostos, considerando todas as instâncias processadas, independentemente dos cenários ao quais estas pertencem, visando avaliar de forma geral o desempenho do método proposto para a diversidade de problemas gerados.

Tabela 5.2 - Piores Casos dos Cenários

Cenário	Tarefas (n)	Embarcações (m)	Fator de Folga (α)	Fator de Liberação (β)	GAP Máximo	GAP + Máximo	GAP - Máximo	GAP 0 Máximo	Número de Iterações Máximo	Tempo de Processamento Máximo (s)
1	20	4	0%	90%	0,25%	0,15%	0,10%	6,74%	1.085	36
2	20	4	50%	90%	0,47%	0,09%	0,38%	21,57%	1.042	36
3	20	4	100%	90%	2,17%	1,23%	1,31%	20,35%	1.732	48
4	20	5	0%	90%	0,05%	0,00%	0,05%	6,29%	1.199	39
5	20	5	50%	90%	0,34%	0,09%	0,28%	22,04%	1.270	37
6	20	5	100%	90%	2,52%	0,22%	2,52%	63,07%	1.358	44
7	30	5	0%	90%	0,48%	0,48%	0,05%	8,16%	1.346	84
8	30	5	50%	90%	0,08%	0,08%	0,04%	8,20%	1.327	65
9	30	5	100%	90%	0,25%	0,05%	0,25%	7,05%	1.099	64
10	30	6	0%	90%	0,04%	0,04%	0,03%	4,70%	1.035	52
11	30	6	50%	90%	0,04%	0,00%	0,04%	10,26%	976	45
12	30	6	100%	90%	3,41%	2,06%	1,42%	14,70%	1.071	62
13	50	6	0%	90%	0,12%	0,12%	0,00%	5,97%	1.173	1.290
14	50	6	50%	90%	0,16%	0,16%	0,03%	9,72%	1.082	1.322
15	50	6	100%	90%	0,19%	0,12%	0,11%	10,47%	1.246	1.316
16	50	8	0%	90%	0,07%	0,07%	0,00%	5,43%	1.038	1.086
17	50	8	50%	90%	0,19%	0,15%	0,08%	10,67%	1.172	1.174
18	50	8	100%	90%	0,63%	0,63%	0,12%	18,57%	1.322	1.361
19	100	8	0%	90%	0,25%	0,25%	0,00%	4,27%	1.428	845
20	100	8	50%	90%	0,15%	0,15%	0,00%	3,76%	1.341	2.686
21	100	8	100%	90%	0,12%	0,12%	0,02%	2,78%	1.571	4.432
22	100	10	0%	90%	0,13%	0,13%	0,00%	4,87%	1.415	3.545
23	100	10	50%	90%	0,14%	0,14%	0,01%	3,37%	1.294	3.327
24	100	10	100%	90%	0,30%	0,29%	0,01%	6,73%	1.322	3.699

Tabela 5.3 - Melhores Casos dos Cenários

Cenário	Tarefas (n)	Embarcações (m)	Fator de Folga (α)	Fator de Liberação (β)	GAP Mínimo	GAP + Mínimo	GAP - Mínimo	GAP 0 Mínimo	Número de Iterações Mínimo	Tempo de Processamento Mínimo (s)
1	20	4	0%	90%	0,00%	0,00%	0,00%	0,44%	86	3
2	20	4	50%	90%	0,00%	0,00%	0,00%	1,58%	234	7
3	20	4	100%	90%	0,00%	0,00%	0,00%	1,46%	149	5
4	20	5	0%	90%	0,00%	0,00%	0,00%	0,44%	131	4
5	20	5	50%	90%	0,00%	0,00%	0,00%	1,41%	98	3
6	20	5	100%	90%	0,00%	0,00%	0,00%	0,00%	203	6
7	30	5	0%	90%	0,00%	0,00%	0,00%	1,16%	227	14
8	30	5	50%	90%	0,00%	0,00%	0,00%	0,91%	278	15
9	30	5	100%	90%	0,00%	0,00%	0,00%	1,71%	305	17
10	30	6	0%	90%	0,00%	0,00%	0,00%	1,31%	301	16
11	30	6	50%	90%	0,00%	0,00%	0,00%	1,75%	263	14
12	30	6	100%	90%	0,00%	0,00%	0,00%	3,66%	358	20
13	50	6	0%	90%	0,00%	0,00%	0,00%	0,76%	356	384
14	50	6	50%	90%	0,00%	0,00%	0,00%	1,83%	357	379
15	50	6	100%	90%	0,00%	0,00%	0,00%	1,78%	780	799
16	50	8	0%	90%	0,00%	0,00%	0,00%	1,10%	466	542
17	50	8	50%	90%	0,00%	0,00%	0,00%	1,77%	406	383
18	50	8	100%	90%	0,00%	0,00%	0,00%	2,86%	466	431
19	100	8	0%	90%	0,00%	0,00%	0,00%	0,77%	650	426
20	100	8	50%	90%	0,01%	0,01%	0,00%	1,00%	895	472
21	100	8	100%	90%	0,00%	0,00%	0,00%	0,94%	493	1.424
22	100	10	0%	90%	0,00%	0,00%	0,00%	0,95%	795	1.925
23	100	10	50%	90%	0,02%	0,02%	0,00%	1,41%	796	2.099
24	100	10	100%	90%	0,02%	0,02%	0,00%	1,89%	942	2.418

Tabela 5.4 - Resumo dos Resultados

	GAP	GAP +	GAP -	GAP 0	Número de Iterações	Tempo de Processamento (s)
Média	0,11%	0,06%	0,05%	4,57%	744	790
Máximo	3,41%	2,06%	2,52%	63,07%	1.732	4.432
Mínimo	0,00%	0,00%	0,00%	0,00%	86	3

Conforme pode ser visto na Tabela 5.4 e nas 3 tabelas que a precedem, o valor médio do indicador GAP^{+} (relação entre o valor da função objetivo da solução ótima do problema e o valor da função objetivo para o limitante superior) e o valor médio do indicador GAP^{-} (relação entre o valor da função objetivo da solução ótima do problema e o valor do limitante inferior), estão muito próximos, assim como o máximo e

mínimo obtidos para esses indicadores. Com base nisto e no fato de que o propósito do método é reduzir ao máximo a diferença entre os limitantes, e se possível zerá-los, as análises feitas abaixo utilizarão apenas o indicador GAP que carrega tanto as informações do GAP+ como do GAP-.

Também foram levantados indicadores do GAP, agrupando os cenários de acordo com o seu porte, ou seja, quantidade de tarefas e quantidade de embarcações disponíveis, como pode ser visto na Tabela 5.5. Como era de se esperar, o aumento do número de tarefas dificulta a resolução do problema. Isso pode ser constatado pelo aumento tanto do número médio de iteraçõe como do tempo médio para resolução das instâncias conforme aumenta o número de tarefas. Em relação à qualidade dos limitantes, aparentemente não há relação com o número de tarefas das instâncias. Independentemente do porte do cenário, o valor médio do GAP variou de 0,0% a 0,2%.

Tabela 5.5 – Resultados agrupados por porte do cenário

Cenários	Tarefas	Embarcações	GAP Médio	GAP Máximo	GAP Mínimo	Número de Iterações Médio	Tempo de Processamento Médio (s)
1 a 3	20	4	0,19%	2,17%	0,00%	446	14
3 a 6	20	5	0,14%	2,52%	0,00%	507	16
7 a 9	30	5	0,04%	0,48%	0,00%	585	34
10 a 12	30	6	0,21%	3,41%	0,00%	577	32
13 a 15	50	6	0,06%	0,19%	0,00%	859	919
16 a 18	50	8	0,08%	0,63%	0,00%	811	812
19 a 21	100	8	0,06%	0,25%	0,00%	1.071	1.514
22 a 24	100	10	0,08%	0,30%	0,00%	1.072	2.903

Observando os resultados da Tabela 5.1, Tabela 5.2 e Tabela 5.3 pode-se notar que os indicadores propostos variam bastante dentro de cenário de mesmo porte (igual número de embarcações e tarefas). Neste caso, a avaliação exclusiva dos cenários agrupados por número de tarefas e número de embarcações não leva em conta todos os aspectos importantes para determinar a dificuldade dos problemas resolvidos.

Para analisar o motivo das diferenças entre cenários de mesmo porte, os resultados serão agrupados de acordo com o parâmetro de dispersão β, lembrando que esse parâmetro representa a dispersão da quantidade de datas disponível para início das tarefas sem que as mesmas incorram em penalidade. A Tabela 5.6 agrupa os resultados de acordo com o parâmetro β.

Conforme pode ser observado na tabela abaixo, o aumento do fator de folga, e o consequente aumento das datas disponíveis para início das tarefas sem que se incorra em penalização é um fator que dificulta a resolução do problema. Os indicadores GAP Médio, GAP Máximo, Número de Iterações e Tempo aumentam conforme o aumento do parâmetro β, ou seja, é esperado que a distância dos limitantes inferior e superior aumente, assim como o esforço computacional necessário, com o aumento de β.

Em relação ao parâmetro α, que impacta na dispersão das datas de liberação, o mesmo foi mantido com o valor constante de 90% para todas as instâncias das tabelas de resultados aqui publicadas. Isto foi feito visando representar problemas reais de programação de frota de embarcações de lançamentos de dutos.

Tabela 5.6 - Resultados agrupados pelo parâmetro (β)

Número de Instâncias	Fator de Folga	GAP Médio	GAP Máximo	GAP Mínimo	Número de Iterações Médio	Tempo de Processamento Médio (s)
80	0%	0,03%	0,48%	0,00%	697	662
80	50%	0,05%	0,47%	0,00%	724	705
80	100%	0,24%	3,41%	0,00%	810	999

Todos os resultados acima apresentados foram obtidos utilizando-se os mesmos critérios de parada, independentemente do número de

tarefas, embarcações ou parâmetros de dispersão. Os critérios de parada utilizados independem do tempo total de processamento, embora isso possa ser modificado futuramente, quando da necessidade de processar cenários de maior porte.

O primeiro critério de parada é atingido quando se encontra uma solução viável de mesmo valor da estimativa da solução viável, zerando o GAP e encontrando a solução ótima do problema.

O segundo e último critério de parada é atingido quando o multiplicador do passo, utilizado na relaxação lagrangiana, inicializado com valor 2, após não melhorar a estimativa da solução ótima por diversas iterações, sofre divisões até ficar menor do 10^{-7}. Os critérios utilizados para atualização do multiplicador de passo foram explicados na seção 4.4.

5.2 **Discussão dos Resultados**

Na seção anterior deste capítulo foram apresentados os resultados obtidos com a aplicação da metodologia proposta no capítulo 4 deste trabalho para a resolução de problemas de PROGRAMAÇÃO DE FROTA DE EMBARCAÇÕES DE LANÇAMENTO DE DUTOS. Abaixo serão discutidos os resultados apresentados.

Antes de iniciar a discussão dos resultados alcançados neste trabalho, é importante destacar que as soluções ótimas e os respectivos valores das funções objetivos foram obtidos utilizando o pacote de otimização CPLEX 12.0. Embora o tempo necessário para resolução dos cenários tenha sido baixo, isso não invalida os resultados desta pesquisa já que o objetivo de determinar limitantes inferior e superior o mais próximos possível, em tempo não proibitivo, foi alcançado com o uso de

ferramentas básicas e grande disponibilidade como é o caso do Microsoft Excel e da linguagem de programação VBA.

Para avaliar o desempenho do método proposto sob diversos tipos de condições, foram geradas várias instâncias de teste, variando a quantidade de recursos disponíveis para atendimento de uma demanda estabelecida. A quantidade de recursos é representada pela disponibilidade de embarcações para atendimento das tarefas. A variação da demanda se dá tanto pela quantidade de tarefas como pela característica dessas tarefas: disponibilidade para início, duração, data de entrega e penalidade associada ao atraso. As características das tarefas por sua vez, são geradas de forma aleatória respeitando os parâmetros de dispersão propostos.

Os limitantes inferiores para os problemas apresentados foram calculadas a partir das soluções relaxadas geradas pelo método da relaxação lagrangiana. A melhor solução viável de cada problema foi gerada pela heurística construtiva, pela heurística que transforma uma solução inviável em viável ou pela busca local. A qualidade das estimativas geradas e das soluções propostas, de forma geral, foram muito boas, dado que o GAP médio de todas as instâncias processadas foi de 0,11% e o GAP máximo foi de 3,41%, como já foi mostrado na Tabela 5.4.

Tabela 5.7 - Faixas de GAP

Nome	Frequência Absoluta	Frequência Relativa	Frequência Relativa Acumulada
GAP = 0	115	47,92%	47,92%
0,00%<GAP<=0,25%	108	45,00%	92,92%
0,25%<GAP<=0,50%	10	4,17%	97,08%
0,50%<GAP<=0,75%	2	0,83%	97,92%
0,75%<GAP<=1,0%	0	0,00%	97,92%
1,0%<GAP<=1,5%	0	0,00%	97,92%
1,5%<GAP<=2,0%	2	0,83%	98,75%
2,0%<GAP<=2,5%	1	0,42%	99,17%
2,5%<GAP<=3,0%	1	0,42%	99,58%

3,0%<GAP<=3,5%	1	0,42%	100,00%

A Tabela 5.7 mostra a freqüência com que foram obtidos GAP's dentro de cada faixa. A Figura 5.1 mostra de forma visual a qualidade dos resultados obtidos.

Pode-se notar pelo que foi exposto acima, que de um total de 240 instâncias testadas, em 97% das mesmas o GAP obtido < 0,5%. Em alguns casos pontuais, 5 instâncias ou pouco mais de 2% do total testado, o GAP obtido foi maior do que 1%.

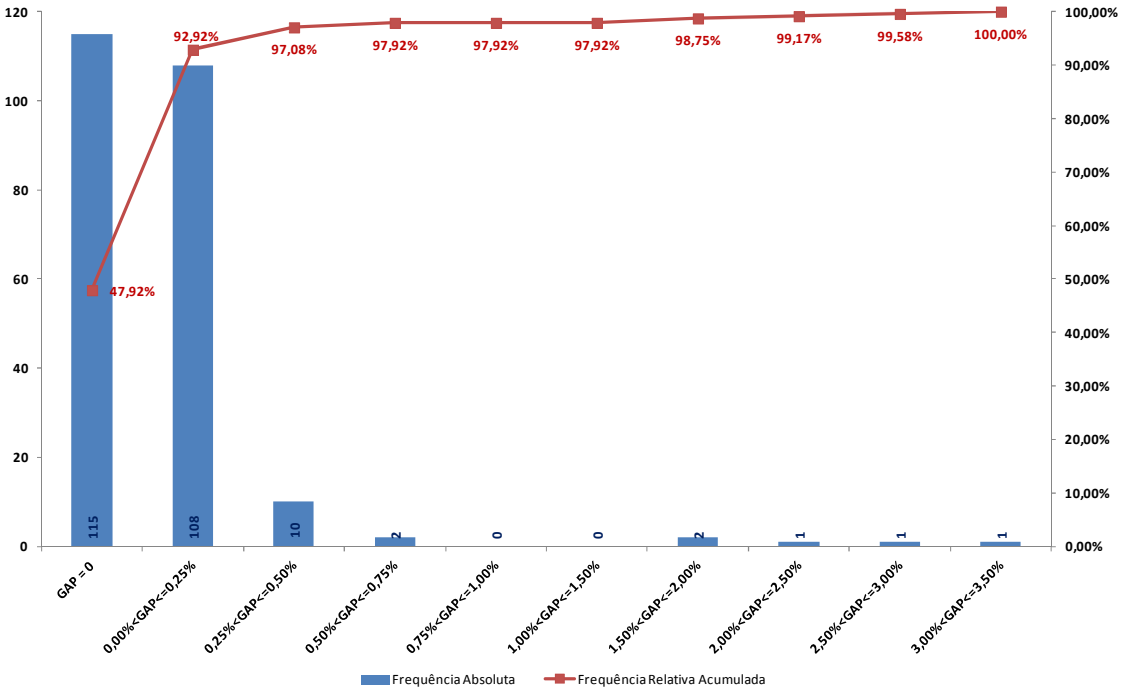

Figura 5.1 – Histograma dos GAPs de todas as instâncias

Tabela 5.8 - Instâncias com GAP > 1%

ID Cenário	ID Instância	Tarefas	Embarcações	Fator de Folga	Fator de Liberação	GAP	GAP +	GAP -	GAP 0	Iterações	Tempo de Processamento (s)
12	114	30	6	100%	90%	3,41%	2,06%	1,42%	9,81%	949	61
6	59	20	5	100%	90%	2,52%	0,00%	2,52%	0,00%	1358	44
3	23	20	4	100%	90%	2,17%	0,88%	1,31%	20,35%	955	28
3	26	20	4	100%	90%	1,87%	1,23%	0,66%	6,55%	1047	36
12	111	30	6	100%	90%	1,84%	0,93%	0,93%	14,33%	1042	61

A Tabela 5.8 foi organizada pela ordem decrescente dos GAP's e traz as instâncias cujo GAP obtido era maior do que 1%. Pode-se notar

que todos estes casos foram obtidos em instâncias cujo parâmetro de dispersão β valia 100%. A relação do aumento deste parâmetro com instâncias cujos limitantes inferiores e superiores estão mais afastados é intuitiva visto que existem mais possibilidades para programação das tarefas. Com exceção da instância 59, na qual o limitante superior coincide com o valor ótimo do problema, o GAP obtido para as demais instâncias foi conseqüência da necessidade de melhorias tanto no limitante inferior como no limitante superior para estas instâncias.

Ainda analisando esta tabela, é possível verificar que as instâncias de maior porte (50 e 100 tarefas) não fazem parte da listagem de instâncias com pior desempenho em termos do tamanho do GAP. Embora essas instâncias de maior porte tenham exigido maior esforço computacional em sua resolução, apresentaram excelentes resultados. Algo que chama atenção em relação às instâncias cujo GAP superou 1% são o baixo tempo de processamento e o alto número de iterações. Isso acontece quando os valores da função objetivo das soluções para o problema relaxado, obtidas durante a execução do algoritmo, estão muito distantes do limitante inferior, implicando na não geração de soluções viáveis e uso da busca local.

Como foi citado na seção 5.1, todas as instâncias foram processadas utilizando os mesmos critérios de parada e parâmetros de calibragem. Caso exista interesse melhorar os resultados de uma instância específica é possível alterar alguns parâmetros de calibragem, como é o caso do parâmetro que define se uma solução inviável será convertida em solução viável. Esse parâmetro pode diminuir o tamanho do GAP obtido com a contrapartida de aumentar o tempo de processamento da instância.

Relembrando que em quase 98% das instâncias testadas o GAP esteve entre 0 e 1%, serão focados os caso em que a metodologia proposta funcionou muito bem. Na Figura 5.2, observa-se a evolução da

melhor solução viável obtida e da estimativa do valor ótimo, a cada iteração, para um problema de 30 tarefas e 5 embarcações. As características das tarefas deste exemplo resolvido estão indicadas na Tabela 5.9.

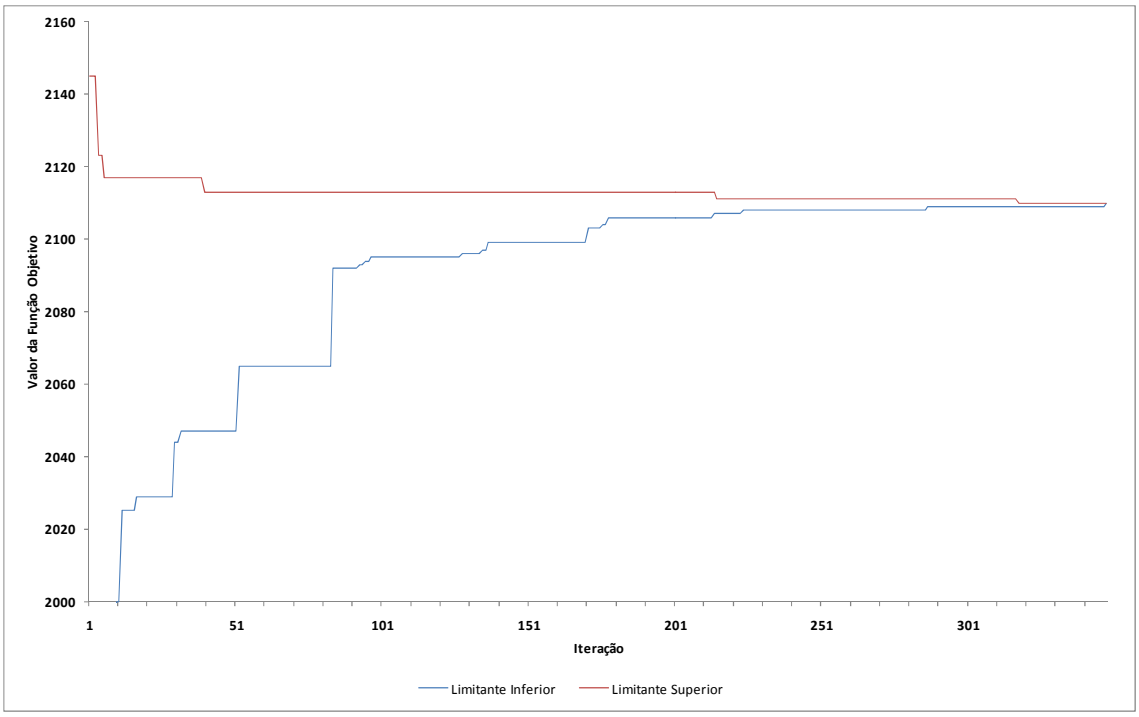

Figura 5.2 - Evolução da Função Objetivo para um problema de 30 tarefas e 5 embarcações

Assim como mostrado acima, a resolução da grande maioria das instâncias resolvidas apresenta a característica de gerar uma solução viável com a heurística construtiva, melhorar bruscamente o valor da melhor solução viável no início das iterações e refinar essa solução enquanto o valor do limitante inferior ainda sobe com muito mais intensidade.

É importante ressaltar que mesmo não fornecendo uma estimativa do valor ótimo logo nas iterações iniciais, as soluções obtidas com o problema relaxado estão muito próximas da solução ótima do problema original, bastando deslocar as tarefas de forma a não mais desrespeitar a restrição que foi relaxada. Após a estabilização da melhor solução viável

encontrada, a estimativa da solução ótima continua a subir principalmente devido aos ajustes nos multiplicadores de Lagrange e não mais devido a grandes mudanças na programação de tarefas.

Tabela 5.9 - Exemplo 30 tarefas e 5 embarcações

Tarefa	Duração	Data de Entrega	Penalidade por Dia de Atraso	Data de Liberação
1	9	10	10	0
2	17	25	9	0
3	6	7	1	0
4	17	21	9	0
5	24	29	5	0
6	4	5	2	0
7	3	4	10	0
8	3	4	1	0
9	8	11	6	0
10	19	26	3	0
11	5	5	4	0
12	18	20	10	0
13	13	13	8	0
14	9	12	2	0
15	25	37	7	0
16	24	30	10	0
17	11	14	2	0
18	5	6	5	0
19	1	1	8	0
20	2	101	2	99
21	27	35	4	0
22	8	12	10	0
23	27	31	9	0
24	6	9	6	0
25	30	41	6	0
26	27	38	2	0
27	18	23	10	0
28	2	3	7	0
29	4	4	2	0
30	28	38	9	0

6 CONCLUSÕES

Neste capítulo serão feitas as conclusões a respeito da metodologia empregada, dos resultados obtidos e serão mencionados possíveis desdobramentos ou futuros desenvolvidos para esta pesquisa, que abrange o problema de PROGRAMAÇÃO DE FROTA DE EMBARCAÇÕES DE LANÇAMENTO DE DUTOS.

Foi estudado o problema de programação de embarcações PLSVs, utilizadas na operação de apoio marítimo de lançamento de dutos submarinos, instalação de risers e interligação de poços. O objetivo da pesquisa é determinar a sequência de atendimento de um conjunto de tarefas de forma que a penalização global pelo atraso na entrada em operação dos poços seja mínima. Embora embarcações do tipo PLSVs sejam embarcações especializadas utilizadas na exploração de petróleo offshore, os métodos utilizados nesta pesquisa podem ser utilizados para resolver problemas de sequenciamento de tarefas em veículos ou máquinas de forma geral.

Para modelar o problema proposto foi apresentada uma formulação indexada ao tempo. A escolha dessa formulação se deu pelo fato de facilitar a aplicação da relaxação lagrangiana, e pelo fato de as soluções relaxadas gerarem boas estimativas da solução ótima para o problema original.

Para resolver os problemas de programação foram propostas: uma heurística construtiva, uma heurística para gerar soluções viáveis a partir de soluções do problema relaxado em soluções viáveis do problema original, e uma heurística de busca local. Além das heurísticas, utilizou-se o método da relaxação Lagrangiana para não só gerar soluções relaxadas, que viriam a ser convertidas em soluções viáveis, mas como

para estimar o limitante inferior da solução ótima encontrada pelos métodos construtivos e de busca.

O escopo inicial do trabalho era resolver problemas de médio porte, da ordem de 100 tarefas e 10 embarcações em um horizonte de planejamento de curto ou médio prazo (3 a 4 meses). Além desse tipo de cenário foram feitos testes com menos tarefas e menos embarcações objetivando-se verificar a confiabilidade e a robustez do método de solução proposto.

Os resultados alcançados para essas instâncias de pequeno e médio porte foram satisfatórios visto que, considerando todas as instâncias testadas, a média do GAP entre a estimativa da solução ótima e a solução viável encontrada foi de apenas 0,11% sendo que em quase 50% dos casos o GAP encontrado foi igual a zero.

Para melhorar os resultados das instâncias em que o limitante inferior ficou relativamente distante do limitante superior, é possível alterar a calibragem de parâmetros, como por exemplo, os parâmetros relacionados a geração de solução viável ou critérios de parada. Além disso, é possível ainda reiniciar a resolução dos problemas substituindo a solução proposta pela heurística construtiva, pela melhor solução viável apontada ao final do método. Fazendo isso, além de aplicar as estruturas de busca local mais vezes, estará sendo fornecido um novo e importante dado de entrada para a relaxação lagrangiana: a estimativa do valor da função objetivo da solução ótima do problema. Essa substituição deve ser o suficiente para reduzir ainda mais o intervalo que separa o limitante inferior do superior.

Um ponto a ser estudado para a melhoria do método que fornece o limitante inferior dos problemas é a atualização do valor do limitante superior ($\bar{L}$, descrito na seção 4.3.4), utilizado como parâmetro no método da relaxação lagrangiana. Atualmente este parâmetro é estimado uma

única vez no início do algoritmo, com a solução encontrada pela heurística construtiva e nunca mais é atualizado. Foram feitos alguns testes para atualização deste parâmetro conforme são geradas soluções de boa qualidade pela heurística de busca local, mas os resultados encontrados não foram satisfatórios. A atualização do parâmetro fazia com que o método da relaxação lagrangiana gerasse soluções de baixa qualidade. Isto pode ser explicado pelo fato da redução do limitante superior implicar na redução do tamanho do passo, o que por sua vez acarreta dificuldade para deslocar a solução do problema relaxado de uma região à outra. Como o método parecia funcionar muito bem sem a atualização deste parâmetro, optou-se, a princípio, por não o fazê-lo.

Existem algumas abordagens disponíveis para tentar melhorar a qualidade das soluções viáveis geradas. Uma delas seria trabalhar na solução gerada pela heurística construtiva utilizando métodos de busca local, antes de iniciar os procedimentos da relaxação lagrangiana. Outra opção seria utilizar diferentes tipos de heurística construtiva para gerar soluções viáveis para o problema e compará-las, a fim de escolher a solução cujo valor da função objetivo fosse o menor. Seriam necessários alguns testes para verificar se esta medida traria vantagens para o método da relaxação lagrangiana no que diz respeito à solução final gerada, a quantidade de iterações necessárias e o tempo de processamento para obtenção das estimativas da solução ótima e das melhores soluções viáveis encontradas.

Outra abordagem seria trabalhar na criação de buscas locais mais elaboradas para tentar melhorar as soluções viáveis encontradas que necessitem de grandes mudanças na estrutura de sua vizinhança, para deixar um ótimo local e atingir o ótimo global do problema. Mais uma vez seriam necessários testes, para avaliar se a implementação de buscas locais mais complexas implicariam em melhora das soluções geradas ou somente em maior tempo de processamento sem ganho de resultado.

O modelo matemático aqui proposto tem suas vantagens, como foi explicado inicialmente, mas sofre com a necessidade de memória, dado o número de variáveis envolvidas no processamento do modelo. Testes preliminares indicam que é possível processar cenários de porte maior como, por exemplo, 150 ou 200 tarefas para uma frota de 15, 20 ou 30 embarcações, embora o tempo de processamento do modelo aumente consideravelmente e em alguns casos se torne proibitivo.

Dada a formulação matemática proposta, um importante fator para o desempenho do algoritmo proposto é o horizonte de planejamento considerado, dado que a quantidade de memória necessária para execução do algoritmo depende diretamente do horizonte. Caso seja possível reduzir a duração das tarefas e os instantes de liberação das mesmas, será possível processar cenários com maior número de tarefas e embarcações sem a necessidade de fazer grandes alterações na metodologia proposta.

A partir dos resultados obtidos e das considerações feitas acima, são resumidas algumas atividades a serem desenvolvidas com o intuito de melhorar as soluções propostas ou então resolver problemas parecidos de maior grandeza:

- Avaliação dos ganhos e perdas de se aplicar algum tipo de busca local mais sofisticada sobre a solução viável criada, a partir da heurística construtiva.
- Implementação de um método para atualização do parâmetro que indica o valor da melhor solução viável, durante a execução da relaxação lagrangiana, visando reduzir o número de iterações para que o método estime a melhor solução viável do problema mais rapidamente ou mesmo melhore essa estimativa.
- Desenvolvimento de outras vizinhanças de busca local visando aumentar o desempenho do algoritmo que encontra soluções viáveis e também melhorar o valor da melhor

solução viável nos casos em que o valor ótimo não foi encontrado.

- Adaptação e melhoria da metodologia de solução proposta visando aumentar as instâncias o tamanho dos cenários resolvidos, possivelmente revendo as características das tarefas e os horizontes de planejamento para que a demanda computacional não seja impeditiva.

7 REFERÊNCIAS BIBLIOGRÁFICAS

AARTS, E.; LENSTRA, J.K. Local Search in combinatorial optimization. Princeton University Press, 2003. 536p.

AKKER J. M. (2000) Time Indexed Formulation for Machine Scheduling Problems: Column Generation, **Journal on Computing** Vol. 12 No.2 Spring 2000

BEASLEY J. E. Extensions to a Lagrangean relaxation approach for the capacitated warehouse location problem (with N.Christofides), **European Journal of Operational Research**, vol.12, p.19-28, 1983.

BERTRAND, J.W.M.; FRANSOO, J.C. Operations management research methodologies using quantitative modeling. **International Journal of Operations & Production Management**, 22:2, p.241-254, 2002.

BODIN, L.D.; GOLDEN B.; ASSAD A.; BALL M. Routing and scheduling of vehicles and crews: The state of the art. **Computers and Operations Research**, 10, p.63-212, 1983

CHAN, L.M.A.; MURIEL, A.; SIMCHI-LEVI D. Parallel Machine Scheduling, Linear Programming, and Parameter List Scheduling Heuristics. **Operations Research**, 46, p.729-741, 1997.

CHENG, T.C.E.;SIN C.C.S. A state-of-the-art review of parallel-machine scheduling research. **European Journal o Operational Research**, 47 p.271-292, 1990

CHRISTIANSEN, M.; FAGERHOLT, K.; RONEN, D. Ship routing and scheduling: Status and Perspectives. **Transportation Science**, 38:1, p.1-18, 2004.

CHRISTIANSEN, M.; FAGERHOLT, K.; NYGREEN, B.; RONEN, D. Maritime Transportation. In: Barnhart, C.; Laporte, G. **Handbook in OR & MS,** 14, p.189-284, 2007.

CHRISTOPHER, B.; PROSSER, P.; SELENSKY, E. Vehicle Routing and Job Shop Scheduling: What's the difference? International Conference on Automated Planning and Scheduling, 2003.

CORDEAU, J.F.; LAPORTE, G.; SAVELSBERGH, M.W.P.; VIGO, D. Barnhart, C.; Laporte, G. Vehicle Routing. In: Transportation **Handbooks in Operations Research and Management Science**, vol. 14. North-Holland: Amsterdam, p.367-428, 2006.

CUNHA, C. B. Uma contribuição para o problema de roteirização de veículos com restrições operacionais. São Paulo: EPUSP, Departamento de Engenharia de Transportes. 222p. (Tese de Doutoramento), 1997.

CUNHA, C. B. Aspectos Práticos da Aplicação de Modelos de Roteirização de Veículos a Problemas Reais. Transportes (ANPET) v.8, 2000.

DE, P.; MORTON, T. E. Scheduling to Minimize Makespan on Unequal Parallel Processors. **Decision Science** 11, p. 586-602, 1980.

DESROSIERS, J.; DUMAS, Y.; SOLOMON, M.M.; SOUMIS, F. Time constrained routing and scheduling. In: Ball M.O.; Magnanti T.L.; Monma C.L.; Nemhauser G.L. (eds). Network Routing: **Handbooks in Operations Research and Management Science**, vol. 8. North-Holland: Amsterdam, p. 35-139, 1995.

ESPEJO L. G. A.; GALVÃO R. D. O Uso das Relaxações Lagrangeana e Surrogate em Problemas de Programação Inteira. **Pesquisa Operacional** v.22 n.3, p. 387-402, 2002.

GLOVER, F. Surrogate constraint duality in mathematical programming. **Operations Research**, 23, p. 434-451, 1975.

GREEBERG, H.J.; PIERSKALLA, W.P. Surrogate mathematical programming. **Operations Research**, 18, p. 1138-1162, 1970

HANSEN, P., & MLADENOVIC, N. A tutorial on variable neighbourhood search. Le cahiers du GERAD G-2003-46, 2003.

JOUGLET, A.; SAVOUREY, D. Dominance rules for the parallel machine total weighted tardiness scheduling problem with release dates. **Computers & Operations Research**, 38, p.1259-1266, 2011

KOUKI, Z.; CHAAR B. F.; HAMMADI S.; KSOURI M. Analogies between Flexible Job Shop Scheduling and Vehicle Routing problems. **Industrial Engineering and Engineering Management,** IEEE International Conference, 2007.

LAPORTE, G.; OSMAN, I.H. Routing problems: a bibliography. **Annals of Operations Research**, 61, p.227-262, 1995.

LAWRENCE S. A. International Sea Transport: The years Ahead. Lexington Books, Lexington, MA, 1972, 316p.

LI, K.; YANG, S. Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations an algorithms. **Applied Mathematical Modelling** v.33 p.2145-2158, 2008

LUH, P.B.; HOITOMT, D.J.; MAX E.; PATTIPATI K.R. Schedule Generation and Reconfiguration for Parallel Machines. **IEEE Transactions on Robotics and Automation**, vol. 6, no. 6, pp. 687-696, 1990.

MENDES, A. B. **Programação de Frota de Apoio a Operações "Offshore" Sujeita a Requisição de Múltiplas Embarcações para uma Mesma Tarefa**. 2007. 240 p. Tese (Doutorado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2007.

MIESNER T.O.; LEFFER W.L. Oil & gas Pipelines In Nontechnical Language. Pennwell, 357p, 2006.

PALMER A.C.; KING R. A. Subsea Pipeline Engineering**.** Pennwell, 575p., 2008.

REEVES C. R. Modern Heuristic Techniques for Combinatorial Problems. John Wiley & Sons, Inc., 320p., 1993.

RITCHIE, G. Practical introduction to anchor handling and supply vessel operations. Houston, Texas: Oilfield Publications Limited, 192p., 2004.

RONEN, D. Cargo ships routing and scheduling: Survey of Models and Problems. **European Journal of Operations Research**, 12 pp. 119-126, 1983.

RONEN, D. Ship scheduling: The last decade. **European Journal of Operations Research** 71(3) pp. 325-333, 1993.

UNIVERSIDADE DE SÃO PAULO. ESCOLA POLITÉCNICA. **Diretrizes para Apresentação de Dissertações e Teses.** 3ª Edição. São Paulo: 2006. 103p.

UNLU, Y.; MASON S. J. Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. **Computers & Industrial Engineering**, 58, pp. 785-800, 2010

WOLSEY, L. **Integer programming**. John Wiley & Sons, Inc., 1998. 266p.