ANDRÉ BERGSTEN MENDES

**PROGRAMAÇÃO DE FROTA DE APOIO A OPERAÇÕES “OFFSHORE”
SUJEITA À REQUISIÇÃO DE MÚLTIPLAS EMBARCAÇÕES PARA UMA
MESMA TAREFA**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Doutor em Engenharia.

São Paulo

2007

ANDRÉ BERGSTEN MENDES

**PROGRAMAÇÃO DE FROTA DE APOIO A OPERAÇÕES “OFFSHORE”
SUJEITA À REQUISIÇÃO DE MÚLTIPLAS EMBARCAÇÕES PARA UMA
MESMA TAREFA**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Doutor em Engenharia.

Área de Concentração:
Engenharia Naval e Oceânica

Orientador:
Prof. Dr. Marco Antonio Brinati

Co-orientador:
Prof. Dr. Miguel Cezar Santoro

São Paulo

2007

FICHA CATALOGRÁFICA

Mendes, André Bergsten

Programação de frota de apoio a operações “offshore” sujeita à requisição de múltiplas embarcações para uma mesma tarefa / A.B. Mendes. -- São Paulo, 2007.

224 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Naval e Oceânica.

1. Operações de transportes 2. Roteirização 3. Modelagem matemática 4. Programação inteira e fluxos em rede I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Naval e Oceânica II. t.

*À Priscila, fonte de amor e inspiração
Ao Gustavo, que inundou nossas vidas de alegria*

AGRADECIMENTOS

Muitas pessoas contribuíram direta e indiretamente para o desenvolvimento desta pesquisa. Gostaria de deixar registrado a minha gratidão a algumas delas: *Prof. Marco Antonio Brinati* – orientador, amigo e incentivador, pela condução segura, ponderada e sobremaneira dedicada a esta pesquisa, e pelo apoio contínuo em todos os momentos; *Prof. Miguel Cezar Santoro* – co-orientador, pelos conhecimentos passados, pelas horas dispendidas em reuniões de acompanhamento, pelo incentivo e empolgação; *Chefia do PNV*, nas pessoas dos professores *Oscar Brito Augusto* e *Hernani Luiz Brinati*, pelo importante apoio durante as etapas da minha formação acadêmica; *Prof. Reinaldo Morabito* e *Prof. Hugo Yoshizaki*, pelas sugestões dadas a este trabalho e pela leitura de versões iniciais deste texto; *Prof. Kazuo Nishimoto* – pela cessão do Laboratório TPN para o processamento dos modelos; *Alunos-monitores Gabriel Von Winckler* e *Marco Aurélio Gomes*, pelo suporte técnico junto ao TPN; *Secretaria do PNV* – Sandra, Lânia, Damares, Adenilson e César, pela amizade e ajuda prestativa; *Ricardo Terumichi Ono* e *Élcio Ribeiro*, colegas de doutoramento, pela amizade e incentivo mútuo; *Família* – pais, avós, irmãos, sogros, cunhados, tios, pelo carinho e apoio permanente; *Priscila* – esposa querida, amorosa e especial, pela cumplicidade, ao vivenciar e compartilhar intensamente todos os momentos, sonhos e conquistas. Por último, sou grato a *DEUS* pela vida, pela presença e ajuda a cada dia.

*“Tudo tem o seu tempo determinado, e há tempo
para todo propósito debaixo do céu:
há tempo de nascer e tempo de morrer;
tempo de plantar e tempo de arrancar o que se plantou;
tempo de matar e tempo de curar;
tempo de derribar e tempo de edificar;
tempo de chorar e tempo de rir;
tempo de prantear e tempo de saltar de alegria;
tempo de espalhar pedras e tempo de ajuntar pedras;
tempo de abraçar e tempo de afastar-se de abraçar;
tempo de buscar e tempo de perder;
tempo de guardar e tempo de deitar fora;
tempo de rasgar e tempo de coser;
tempo de estar calado e tempo de falar;
tempo de amar e tempo de aborrecer;
tempo de guerra e tempo de paz.”*

Eclesiastes de Salomão

RESUMO

A presente pesquisa aborda um problema de roteirização e programação de veículos incorporando uma nova restrição operacional: a requisição simultânea de múltiplos veículos para atendimento da demanda. Trata-se de uma característica encontrada em operações de apoio à exploração de petróleo “offshore”, em que mais de uma embarcação é requerida para executar tarefas de reboque e lançamento de linhas de ancoragem. Esta imposição, somada às restrições de janela de tempo, precedência entre tarefas, autonomia das embarcações e atendimento integral da demanda, configuram este problema. A programação é orientada pela minimização dos custos variáveis da operação e dos custos associados ao nível de serviço no atendimento. Este problema é uma variação do problema clássico de roteirização e programação de veículos com janela de tempo, de classe *NP-Difícil*. Nesta pesquisa, propõe-se modelar e resolver o problema em escala real por meio do algoritmo “branch and cut” acoplado às heurísticas de busca em vizinhança “local branching” e “variable neighborhood search”. Para gerar as soluções iniciais será empregado o método “feasibility pump” e uma heurística construtiva.

Palavras-chave: Roteirização e Programação de veículos. Programação matemática. Programação inteira mista.

ABSTRACT

This research focuses a fleet scheduling problem with new operational constraints: each task requiring multiple types of vehicles simultaneously. This kind of operation occurs in offshore exploitation and production sites, when more than one vessel is needed to accomplish the tugging and mooring of oil platforms. Other constraints are maintained such as time windows, precedence between tasks, route duration and the demand attendance. The solution schedules are cost oriented, which encompasses the routing variable costs and the customer service costs. This is a variation of the classical fleet routing and scheduling, which is an NP-Hard problem. This research aims to solve the real scale problem through a combined use of branch and cut strategy with local search algorithms such as local branching and variable neighborhood search. An efficient heuristic rule will be used in order to generate initial solutions using the feasibility pump method.

Keywords: Vehicle routing and scheduling. Mathematical Programming. Mixed-Integer Programming Problems.

SUMÁRIO

1	INTRODUÇÃO	1
1.1	EXPLORAÇÃO DE PETRÓLEO “OFFSHORE”	2
1.2	DETALHAMENTO DO PROBLEMA	5
1.3	MOTIVAÇÃO DO TEMA	10
1.4	OBJETIVOS & METODOLOGIA	11
1.5	DELINEAMENTO DO TEXTO	13
2	REVISÃO BIBLIOGRÁFICA.....	15
2.1	A INDÚSTRIA DE TRANSPORTE MARÍTIMO	15
2.2	INTRODUÇÃO À TAXONOMIA DO PROBLEMA DE PESQUISA.....	23
2.3	MÉTODOS DE SOLUÇÃO EMPREGADOS EM PROBLEMAS DE ROTEIRIZAÇÃO E PROGRAMAÇÃO DE VEÍCULOS	29
2.4	MÉTODOS DE SOLUÇÃO EMPREGADOS EM PROBLEMAS DE PROGRAMAÇÃO LINEAR INTEIRA MISTA	33
2.4.1	<i>Branch and Bound</i>	34
2.4.2	<i>Reformulação</i>	37
2.4.2.1	Desigualdade Válida de Chvátal-Gomory (Gomory’s Fractional Cutting Plane Algorithm)	40
2.4.2.2	Desigualdade Válida Básica para Programação Inteira Mista (The Basic Mixed Integer Inequality)	43
2.4.2.3	Desigualdade Válida para Arredondamento em Programação Inteira Mista (The Mixed Integer Rounding Inequality)	43
2.4.2.4	Desigualdades Disjuntivas (Disjunctive Inequalities).....	44
2.4.2.5	Desigualdades de Cobertura (Cover Cuts).....	44
2.4.2.6	Cortes de Limitantes Implícitos (Implied Bounds Cuts).....	45
2.4.2.7	Cortes por Incompatibilidade (Clique Cuts)	45
2.4.2.8	Desigualdades de Cobertura de Fluxo (Flow Cover Inequalities).....	46
2.4.2.9	Desigualdade por Cobertura de Caminhos de Fluxo (Flow Path Cuts).....	47
2.4.2.10	Desigualdade de Cobertura GUB (GUB Cuts)	47
2.4.3	<i>Branch and Cut</i>	47
2.4.4	<i>Heurísticas de Construção</i>	49
2.4.4.1	LP-and-Fix.....	51
2.4.4.2	Relax-and-Fix	51
2.4.4.3	Feasibility Pump.....	53

2.4.5	<i>Heurísticas de Melhoria</i>	55
2.4.5.1	Relaxation Induced Neighborhood Search – “RINS”	55
2.4.5.2	Local Branching.....	56
2.4.5.3	Variable Neighborhood Search	58
2.5	CONCLUSÃO DA REVISÃO BIBLIOGRÁFICA	60
3	MODELAGEM MATEMÁTICA	63
3.1	DESCRIÇÃO DETALHADA DO PROBLEMA	63
3.2	ESTRATÉGIAS DE MODELAGEM DO PROBLEMA DE PROGRAMAÇÃO DE FROTA	65
3.3	FORMULAÇÃO MATEMÁTICA DE FLUXO EM REDE	66
3.3.1	<i>Nós da rede</i>	66
3.3.2	<i>Parâmetros referentes às tarefas</i>	67
3.3.3	<i>Arcos da rede</i>	67
3.3.4	<i>Parâmetros referentes às embarcações</i>	68
3.3.5	<i>Parâmetros referentes ao binômio tarefa - embarcação</i>	70
3.3.6	<i>Duração das viagens</i>	71
3.3.7	<i>Modelo de “Fluxo em Rede – Completo”</i>	72
3.3.8	<i>Modelo de “Fluxo em Rede – Compacto”</i>	75
3.3.9	<i>Extensões do Modelo</i>	77
3.3.9.1	Afretamento	78
3.3.9.2	Abastecimento & Troca de Tripulação	80
3.3.9.3	Agrupamento em Conjunto de Embarcações.....	81
3.4	MODELO DE PROGRAMAÇÃO DE FROTA EM ESCALA DE TEMPO DISCRETIZADA.....	85
3.4.1	<i>Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão I</i>	86
3.4.2	<i>Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão II</i>	89
3.4.3	<i>Modelo de Programação de Frota em Escala de Tempo Discretizada, com Deslocamento</i>	93
3.4.4	<i>Extensões do Modelo</i>	98
3.4.4.1	Decisões de Afretamento.....	98
3.4.4.2	Abastecimento e Troca de Tripulação	99
3.4.4.3	Agrupamento em Conjunto de Embarcações.....	100
3.5	CONCLUSÃO	101
4	MÉTODOS DE SOLUÇÃO	102
4.1	CONSIDERAÇÕES PRELIMINARES.....	102

4.2	HEURÍSTICA CONSTRUTIVA (H&C)	104
4.2.1	<i>Parâmetros e variáveis da heurística</i>	109
4.2.2	<i>Sorteio do conjunto de embarcações & diferenciação entre as versões básica e gulosa da heurística construtiva</i>	111
4.2.3	<i>Sorteio da tarefa</i>	112
4.3	“FEASIBILITY PUMP”	116
4.4	“LOCAL BRANCHING” (LB)	120
4.5	“VARIABLE NEIGHBORHOOD SEARCH” (VNS)	123
4.6	CONCLUSÃO	126
5	APLICAÇÃO DOS MÉTODOS DE SOLUÇÃO	127
5.1	BASE DE DADOS	129
5.2	“BRANCH & CUT” APLICADO AO MODELO DE FLUXO EM REDE – COMPLETO (BC FLX COMPLETO)	133
5.3	“BRANCH & CUT” APLICADO AO MODELO DE FLUXO EM REDE – COMPACTO (BC FLX COMPACTO)	136
5.4	HEURÍSTICA CONSTRUTIVA – VERSÃO BÁSICA (HC BÁSICA).....	148
5.5	HEURÍSTICA CONSTRUTIVA – VERSÃO GULOSA (HC GULOSA)	156
5.6	HEURÍSTICA CONSTRUTIVA & “BRANCH AND CUT”	160
5.7	HEURÍSTICA CONSTRUTIVA & “VARIABLE NEIGHBORHOOD SEARCH”	164
5.8	HEURÍSTICA CONSTRUTIVA & “LOCAL BRANCHING”	168
5.9	INFLUÊNCIA DAS TAREFAS DE ABASTECIMENTO E TROCA DE TRIPULAÇÃO.....	171
5.10	INFLUÊNCIA DA PARCELA DE PENALIDADE	176
5.11	INFLUÊNCIA DAS DESIGUALDADES VÁLIDAS	177
5.12	“BRANCH AND CUT” APLICADO AO MODELO DE PROGRAMAÇÃO DE FROTA EM ESCALA DE TEMPO DISCRETIZADA, SEM DESLOCAMENTO – VERSÃO II.....	180
5.13	“BRANCH AND CUT” APLICADO AO MODELO DE PROGRAMAÇÃO DE FROTA EM ESCALA DE TEMPO DISCRETIZADA, COM DESLOCAMENTO.....	189
5.14	“FEASIBILITY PUMP” APLICADO AO MODELO DE FLUXO EM REDE - COMPACTO	193
5.15	“FEASIBILITY PUMP” APLICADO AO MODELO DE PROGRAMAÇÃO DE FROTA EM ESCALA DE TEMPO DISCRETIZADA, SEM DESLOCAMENTO – VERSÃO II	195
5.16	DISCUSSÃO DOS RESULTADOS.....	197
6	CONCLUSÕES E RECOMENDAÇÕES.....	211
7	REFERÊNCIAS BIBLIOGRÁFICAS.....	217

LISTA DE ILUSTRAÇÕES

Figura 1.1 – Embarcação tipo AHTS	6
Figura 1.2 – Simulação da operação de lançamento de âncora (Augusto et al., 2002)	7
Figura 1.3 – Metodologia para solução de problemas em Pesquisa Operacional (adaptado de Bertrand; Fransoo, 2002).....	12
Figura 2.1 – Pseudo código do algoritmo “Branch and Bound” (adaptado de Pochet; Wolsey, 2006)...	37
Figura 2.2 – Pseudo código do algoritmo de Planos de Corte (adaptado de Pochet; Wolsey, 2006)	49
Figura 2.3 – Pseudo código do algoritmo “Branch and Cut” (adaptado de Pochet; Wolsey, 2006).....	50
Figura 2.4 – Estrutura geral da busca em vizinhança variável (adaptado de Hansen; Mladenović, 2001)	59
Figura 2.5 – Estrutura geral da busca descendente em vizinhança variável (adaptado de Hansen; Mladenović, 2001)	59
Figura 4.1 – Estrutura geral da heurística construtiva	106
Figura 4.2 – Fluxograma do procedimento de escolha das regras de decisão	113
Figura 4.3 – Pseudo código da heurística construtiva	116
Figura 4.4 – Pseudo código do algoritmo “Feasibility Pump” (adaptado de Achterberg; Berthold, 2005)	118
Figura 4.5 – Pseudo código do algoritmo “Local Branching” (adaptado de Fischetti; Lodi, 2003).....	121
Figura 4.6 – Pseudo código do algoritmo “VNS” (adaptado de Hansen, et al., 2006)	125
Figura 5.1 – Comparação do desempenho do método “branch and cut” para o cenário #20, aplicando-o às formulações completa e compacta do modelo de fluxo em rede.....	139
Figura 5.2 – Gráfico de Gantt das tarefas – cenário #5.....	145
Figura 5.3— Gráfico de Gantt das tarefas com fluxo de embarcações – cenário #5.....	146
Figura 5.4— Gráfico de Gantt das embarcações contendo a sequência de tarefas realizadas – cenário #5	147
Figura 5.5 – Evolução da heurística construtiva – versão básica, aplicada ao cenário #5.....	153
Figura 5.6 - Evolução da heurística construtiva – versão gulosa, aplicada ao cenário #14.....	159
Figura 5.7 – Evolução do método “branch and cut” aplicado à solução inicial heurística, para o cenário #17	164
Figura 5.8 - Evolução do método “variable neighborhood search” aplicado à solução inicial heurística, para o cenário #18	167

Figura 5.9 - Evolução do método “local branching” aplicado a partir de uma solução heurística inicial, para o cenário #18	171
Figura 5.10 - Gráfico de Gantt das tarefas – cenário #5	187
Figura 5.11 – Gráfico de Gantt das embarcações contendo a seqüência de tarefas realizadas – cenário #5	188
Figura 5.12 – Comparação de desempenho do “branch and cut” aplicado ao cenário #13 com e sem solução inicial	201
Figura 5.13 – Comparação de desempenho do “branch and cut” aplicado ao cenário #22 com e sem solução inicial	201
Figura 5.14 – Comparação de desempenho do “branch and cut” aplicado ao cenário #27 com e sem solução inicial	202

LISTA DE TABELAS

Tabela 1.1 – Tipos de Unidades de Perfuração - adaptação de Dyke (1997)	4
Tabela 2.1 – Características dos Problemas de Roteirização (Bodin et al., 1983)	24
Tabela 2.2 - Taxonomia de Desrochers et al. (1990)	27
Tabela 3.1 – Exemplo de Conjuntos de Embarcações	82
Tabela 3.2 – Exemplo de Utilização da Matriz M	92
Tabela 4.1 – Lista de parâmetros e variáveis utilizados no loop principal da heurística construtiva...	106
Tabela 4.2 - Lista de parâmetros e variáveis utilizados no processo de decisão da heurística construtiva	107
Tabela 4.3 - Lista de parâmetros e variáveis utilizados no “feasibility pump”	117
Tabela 4.4 - Lista de parâmetros e variáveis utilizados no “local branching”	120
Tabela 4.5 - Lista de parâmetros e variáveis utilizados no VNS	124
Tabela 5.1 - Métodos de Solução e Correspondentes Modelos Matemáticos	128
Tabela 5.2 – Parâmetros dos cenários	132
Tabela 5.3 – Resultados do método “branch and cut” aplicado ao modelo de fluxo em rede – formulação completa	134
Tabela 5.4 – Resultados agrupados por número de tarefas	135
Tabela 5.5 - Resultados do método “branch and cut” aplicado ao modelo de fluxo em rede – formulação compacta	137
Tabela 5.6 - Resultados agrupados por número de tarefas.....	138
Tabela 5.7 – Dados das tarefas – cenário #5	139
Tabela 5.8 – Dados das embarcações – cenário #5.....	140
Tabela 5.9 – Dados dos locais – cenário #5	141
Tabela 5.10 – Resultados do método “branch and cut” – cenário #5.....	141
Tabela 5.11 – Fluxo das embarcações – cenário #5	143
Tabela 5.12 – Nível de serviço da solução – cenário #5	144
Tabela 5.13 – Parâmetros testados na heurística construtiva – versão básica	150
Tabela 5.14 – Resultados da heurística construtiva – versão básica	151
Tabela 5.15 – Parâmetros da heurística construtiva – versão básica, da melhor solução de cada cenário	154
Tabela 5.16 – Parâmetros usados nas soluções de menor custo	155
Tabela 5.17 - Parâmetros testados na heurística construtiva – versão gulosa	157
Tabela 5.18 - Resultados da heurística construtiva – versão gulosa	158

Tabela 5.19 – Parâmetros da heurística construtiva – versão gulosa, da melhor solução de cada cenário	161
Tabela 5.20 – Parâmetros usados nas soluções de menor custo	162
Tabela 5.21 - Resultados do método “branch and cut” aplicado ao modelo de fluxo em rede – formulação compacta, com solução inicial heurística.....	163
Tabela 5.22 - Resultados do método “variable neighborhood search” aplicado ao modelo de fluxo em rede – formulação compacta, com solução inicial heurística	166
Tabela 5.23 - Resultados do método “local branching” aplicado ao modelo de fluxo em rede – formulação compacta, com solução inicial heurística.....	169
Tabela 5.24 – Lista de Tarefas do Cenário #29	173
Tabela 5.25 – Comparação dos Modelos com e sem Abastecimento para o Cenário #29	174
Tabela 5.26 – Influência das Tarefas de Abastecimento nos Cenários #25 a #28.....	175
Tabela 5.27 – Influência da Parcela de Penalidade para o Cenário #11.....	177
Tabela 5.28 – Influência da Parcela de Penalidade para os Cenários #1, #3 e #7.....	178
Tabela 5.29 – Configurações de Desigualdades Testadas	180
Tabela 5.30 – Testes com as Desigualdades Válidas	181
Tabela 5.31 - “Branch and cut” aplicado ao Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão II.....	183
Tabela 5.32 - Resultados agrupados por número de tarefas.....	184
Tabela 5.33 – Especificação das variáveis binárias com valor igual a 1 na melhor solução gerada pelo método “branch and cut” para o cenário #5.....	185
Tabela 5.34 – Possíveis instantes de início para as tarefas do cenário #5.....	185
Tabela 5.35 – Instantes de início para as tarefas do cenário #5.....	186
Tabela 5.36 – Comparação da solução para o cenário #5, para os modelos de fluxo em rede e de programação de frota em escala de tempo discretizada.....	189
Tabela 5.37 – Resultados do método “branch and cut” aplicado ao modelo de programação de frota em escala de tempo discretizada – fator de discretização 3	191
Tabela 5.38 - Resultados do método “branch and cut” aplicado ao modelo de programação de frota em escala de tempo discretizada – fator de discretização 4	191
Tabela 5.39 - Resultados do método “branch and cut” aplicado ao modelo de programação de frota em escala de tempo discretizada – fator de discretização 5	191
Tabela 5.40 - Resultados agrupados por fator de discretização	192
Tabela 5.41 – Detalhamento das soluções, por fator de discretização, para o modelo de programação de frota em escala tempo discretizada, para o cenário #5	192

Tabela 5.42 – Parâmetros usados no método “feasibility pump”	193
Tabela 5.43 – Resultados do método “feasibility pump” aplicado ao modelo de fluxo em rede – versão compacta	194
Tabela 5.44 – Resultados do método “feasibility pump” aplicado ao modelo de programação de frota em escala de tempo discretizada, sem deslocamento - II	196
Tabela 5.45 – Heurística construtiva processada por 12 horas	199
Tabela 5.46 – Tabela Resumo – Valor da função objetivo da melhor solução gerada pelos diversos métodos utilizados.....	205
Tabela 5.47 – Comparação entre os métodos de solução quanto ao valor da função objetivo da melhor solução gerada.....	207
Tabela 5.48 – Comparação entre os métodos de solução	208

1 INTRODUÇÃO

Este capítulo tem for finalidade introduzir o problema abordado nesta pesquisa, intitulada PROGRAMAÇÃO DE FROTA DE APOIO A OPERAÇÕES “OFFSHORE” SUJEITA À REQUISICÃO DE MÚLTIPLAS EMBARCAÇÕES PARA UMA MESMA TAREFA, definir os objetivos pretendidos, apresentar a metodologia utilizada e o encaminhamento proposto.

A pesquisa, envolvendo a definição de roteiros de veículos e contemplando os aspectos temporais das viagens e as restrições de janelas de tempo para o início do atendimento dos clientes, entre outras imposições, é conhecida como o estudo dos problemas de roteirização e programação de veículos (PRPVs).

A comunidade acadêmica, atuante na área da Pesquisa Operacional, há tempo tem pautado seus esforços na resolução de problemas complexos em cadeias de suprimento, dando ênfase particular à classe de problemas de PRPVs (Desrosiers et al., 1995).

A complexidade destes problemas advém da realidade observada nas operações de transportes, que é complexa e impõe diversos condicionantes a quem programa os veículos, como durações dos roteiros, limites de carga por veículo, compatibilidade veículo – cliente, entre outros (Bodin et al., 1983). Somada a estas restrições, tem-se o número de possibilidades de solução que, por se tratar de um problema combinatório, cresce de maneira não-polinomial com a ordem do problema (Laporte; Osman, 1995).

A compreensão do problema abordado nesta pesquisa requer o detalhamento da operação de transporte em questão, juntamente com as premissas adotadas.

1.1 Exploração de Petróleo “Offshore”

A atividade de exploração de petróleo em alto mar, conhecida como sistema de exploração “offshore”, foi fortemente impulsionada pela descoberta das reservas do Mar do Norte em 1969, quando o cenário mundial era caracterizado por uma elevada e crescente dependência dos países desenvolvidos em relação ao petróleo oriundo dos países membros da OPEP¹. A primeira crise do petróleo em 1973 corroborou para a aceleração do processo de desenvolvimento tecnológico de exploração “offshore”.

A tecnologia desenvolvida para lidar com o ambiente de exploração “offshore” baseava-se em duas estratégias: manter os equipamentos em terrenos firmes ou deslocá-los às áreas alagadas. Neste caso, foram utilizadas desde aterros até embarcações, estruturas flutuantes e estruturas marítimas fixas.

Esta segunda estratégia levou à criação de ilhas artificiais ou plataformas de trabalho nas quais pudessem ser instalados os equipamentos de perfuração, coleta e transporte do petróleo. Isto passou a ser ainda mais necessário com o distanciamento dos poços de petróleo em relação à costa, requerendo infra-estruturas complexas e auto-suficientes, capazes de tornar as unidades de produção independentes.

¹ OPEP - Organização dos Países Exportadores de Petróleo - foi fundada em 1960 pelo Irã, Iraque, Kuwait, Arábia Saudita e Venezuela. Posteriormente, filiaram-se o Catar (1961), Indonésia (1962), Líbia (1962), União dos Emirados Árabes (1967), Argélia (1969), Nigéria (1971), Equador (1973-1992) e Gabão (1975-1994). O seu objetivo é coordenar e unificar as políticas de preço, definir cotas de produção para os países membros, visando garantir uma oferta regular de petróleo que atenda aos seus interesses (políticos e econômicos).

Até chegar às estruturas flutuantes atualmente empregadas, a evolução tecnológica pautou-se nas características de baixa lâmina d'água² dos poços, criando estruturas fixas que se apoiavam no fundo do mar, tais como as plataformas submersíveis e as plataformas auto-elevatórias ou “jack-ups”, que são embarcações dotadas de pilares retráteis acionadas por macacos hidráulicos. Estas, ao tocarem o fundo do mar, tinham as suas estruturas erguidas acima do nível do mar, ficando livres dos efeitos de ondas e correntezas. As estruturas auto-elevatórias tinham a vantagem de poderem se deslocar para outras áreas quando necessário, apenas retraíndo os pilares de sustentação.

Com o aumento da lâmina d'água dos poços de petróleo, as estruturas fixas deixaram de ser viáveis devido ao custo da estrutura crescer exponencialmente com a profundidade. O mesmo acontece para as estruturas auto-elevatórias que deixam de ser viáveis para profundidades superiores a 200 metros. Novos conceitos de embarcações e estruturas flutuantes para a exploração e produção de petróleo surgiram, como os navios-sonda e as plataformas semi-submersíveis de perfuração usados na exploração, e as plataformas semi-submersíveis para a produção.

As estruturas de produção têm como principal característica serem estáticas, uma vez que irão comportar todos os equipamentos necessários para a produção de petróleo por um período correspondente à vida útil dos poços cobertos por aquela plataforma. Após este período, que pode durar vários anos, as estruturas de produção podem, eventualmente, ser alocadas a outros poços, passando antes por um período de manutenção e substituição de equipamentos.

Os navios-sonda e as plataformas semi-submersíveis de perfuração “têm como principal característica a mobilidade para permitir-lhes trabalhar em diferentes

² Termo que indica a profundidade do leito marinho em relação ao nível do mar.

poços. Podem ser usados em várias fases da vida de um campo; na exploração, perfurando poços exploratórios e delimitadores; na fase de desenvolvimento do campo, perfurando e completando os poços que irão drenar o reservatório de acordo com o projeto de exploração; durante a fase produtiva do campo, permitindo dar manutenção aos poços em produção (troca de equipamentos internos, mudanças de zonas de produção, etc.) e, finalmente, no final da vida de cada poço, estes têm que ter seus equipamentos retirados e condicionados para o abandono definitivo.”³

A tabela 1.1, compilada a partir de Dyke (1997), mostra os diferentes tipos de unidades de perfuração e suas características mais relevantes, como: máxima lâmina d’água de perfuração; a forma de fixação, se esta ocorre apoiando a unidade no leito marinho ou se a mesma é flutuante; se a unidade é móvel, ou seja, se ela é empregada em sucessivos locais; e se a unidade é dotada de sistema de propulsão.

Tabela 1.1 – Tipos de Unidades de Perfuração - adaptação de Dyke (1997)

Unidade de Perfuração	Máx. Lâmina d'Água [m]	Fixação	Móvel	Auto-Propelida
Plataforma Submersível	45,0	Apoiada no Leito Marinho	Não	Não
Plataforma Auto-elevatória	200,0	Apoiada no Leito Marinho	Sim	Não*
Barcaça	30,0	Flutuante; Ancorada	Sim	Não
Navio Sonda	Águas Profundas	Flutuante; Ancorado ou Sistema de Posicionamento Dinâmico	Sim	Sim
Plataforma Semi-Submersível	Águas Profundas	Flutuante; Ancorada	Sim	Não*
Plataforma Rígida	300,0	Apoiada no Leito Marinho	Não	Não
Plataforma Complacente	Águas Profundas	Apoiada no Leito Marinho	Não	Não

* Algumas plataformas possuem sistemas de auto-propulsão

³ NAGLE, 2001, Capítulo III, página 54

De particular interesse para o escopo deste trabalho, são as unidades móveis de perfuração empregadas em águas profundas, desprovidas de sistema propulsivo, ou seja, as plataformas semi-submersíveis; também são de interesse as plataformas semi-submersíveis de produção, ambas utilizadas na costa do Brasil.

1.2 Detalhamento do Problema

O tema abordado nesta pesquisa está centrado nas tarefas de reposicionamento e ancoragem das plataformas semi-submersíveis utilizadas para perfuração e produção. Além da alocação de plataformas de perfuração na manutenção dos poços em atividade, uma empresa petrolífera utiliza seus recursos produtivos em um programa contínuo de perfuração em novas áreas e naquelas já mapeadas por procedimentos geológicos. A demanda por deslocamento e ancoragem destas plataformas concorre com as tarefas de reboque e ancoragem das unidades de produção, que entram em funcionamento após o término da perfuração e completação dos poços, ou seja, após o término da preparação dos poços para a etapa produtiva. Essas unidades de produção, em geral, são plataformas semi-submersíveis ou sistemas tipo FPSO⁴, que necessitam ser rebocados a partir de estaleiros onde foram finalizados e ancorados em seus locais de produção.

A questão central a ser resolvida nesta pesquisa diz respeito à programação das embarcações de apoio que realizam o reboque e a ancoragem das plataformas semi-submersíveis de perfuração e de produção.

⁴ FPSO ou “Floating Production Storage and Offloading” é um tipo de navio utilizado pela indústria petrolífera para armazenar o petróleo e/ou gás natural produzido até que um navio aliviador venha escoar a produção. Oleodutos diretamente ligados ao FPSO também podem ser utilizados para o escoamento da produção.

A demanda por estes serviços é atendida por embarcações especializadas denominadas AHTS (“Anchor Handling Tug Supply”). Estes rebocadores possuem como principais características: alta capacidade de tração estática, variando entre 30 e 300 toneladas métricas⁵; elevada potência do motor principal, podendo chegar até 27.000 BHP; área de convés necessária para armazenar as âncoras, os cabos e os suprimentos diversos, variando de 300 a 500 m² e podendo comportar, em peso, 1.000 toneladas em equipamentos (embarcações maiores suportam até 4.000 toneladas). Além disso, dispõem de guinchos necessários para lançamento dos cabos utilizados na processo de ancoragem (Ritchie, 2004). A figura 1.1 mostra uma embarcação tipo AHTS.

Figura 1.1 – Embarcação tipo AHTS

É importante destacar que o processo de reboque e ancoragem de uma plataforma de perfuração ou produção poderá requerer mais de uma embarcação tipo AHTS. No caso do reboque, isto será função do porte da plataforma, da distância

⁵ Potência de tração estática ou “Bollard Pull” é um critério de projeto tradicional usado para a escolha de hélices para rebocadores. É a propulsão teórica atingida a uma velocidade de avanço zero e plena rotação do motor. Para rebocar um navio com 160.000 toneladas de porte bruto, por exemplo, são necessários três rebocadores com potência de tração estática de 80 toneladas cada (Pilots, 2007).

a ser percorrida e das condições de navegação do percurso. Múltiplas embarcações poderão ser necessárias para garantir as condições mínimas de segurança e desempenho.

Com relação ao lançamento das linhas de ancoragem, este consiste no içamento, por parte do rebocador, de uma linha de ancoragem de uma plataforma presa a uma âncora, com o auxílio de um cabo de apoio ("work wire"). À medida que o rebocador se distancia da plataforma, tanto o guincho da plataforma como do rebocador liberam quantidades restritas, respectivamente, de linha de ancoragem e de "work wire" até que a âncora atinja o leito marinho na posição desejada, como mostra a figura 1.2.

Figura 1.2 – Simulação da operação de lançamento de âncora (Augusto et al., 2002)

A necessidade de manter o correto posicionamento da plataforma durante a operação de ancoragem associada ao fato que, o peso da linha, em grandes lâminas d'água, gera elevados esforços no cabo de apoio, obriga que múltiplas embarcações sejam designadas simultaneamente para a realização destas operações.

Programar a frota de AHTS consiste em determinar como cada embarcação será empregada, isto é, qual a seqüência de tarefas, de um conjunto de requisições previamente definido, que cada uma deverá cumprir. Neste processo, algumas restrições operacionais devem ser observadas, como a janela de tempo para início das tarefas e a quantidade mínima de embarcações requeridas por classe. É importante destacar a restrição de duração máxima do roteiro, que reflete a periodicidade de troca da tripulação da embarcação. Tendo em vista que as regiões sendo perfuradas e entrando em operação estão cada vez mais distantes da costa, em águas profundas e ultra-profundas, estas embarcações concentram suas atividades em regiões próximas entre si, afastadas, contudo, da base operacional.

Uma empresa petrolífera utiliza critérios econômicos para priorizar ou sequenciar o atendimento das tarefas. A função de custo total da operação, que deve ser minimizada, contempla a parcela de custo variável da frota, que é consequência do consumo de combustível das embarcações ao se reposicionarem, e de um indicador econômico que traduz a importância do serviço. Tratando-se do reboque e ancoragem de plataformas de produção, o atraso na entrada em operação daquela plataforma adia o retorno ou remuneração do investimento realizado pela empresa petrolífera, que é proporcional ao valor associado ao potencial de exploração daquela plataforma, em seu estágio inicial de operação. A penalização, no caso das plataformas de produção, é certamente maior que em regiões que estão sendo perfuradas em caráter exploratório ou preparatório para a etapa produtiva. A parcela de penalização pode, portanto, ser compreendida como uma parcela associada ao nível de serviço, uma vez que a sua incidência ocorre em função do atraso para o início das tarefas de reboque e ancoragem.

O problema de programação pressupõe que a frota já está devidamente caracterizada, ou seja, a incorporação ou dispensa de embarcações da frota não faz parte dos objetivos do problema.

Em geral, uma empresa de petróleo opta por afretar (contratar) estas embarcações por períodos de médio ou longo prazo, em contratos do tipo “time charter”. Nesta modalidade de afretamento, o armador arca com os custos fixos da embarcação, entre eles: registro da embarcação, seguro do casco & máquina, seguro contra acidentes, manutenções, docagens, inspeções, salários e encargos da tripulação. O afretador, por sua vez, remunera o armador pagando o “hire”, isto é, uma taxa diária de afretamento, definida em contrato, e arca com os custos variáveis (combustível, lubrificantes e taxas portuárias).

A demanda é expressa por um conjunto de requisições de serviço que uma empresa prevê em seu planejamento anual para um determinado horizonte. Caracterizar a demanda para períodos futuros é um exercício complexo, em função das incertezas inerentes à determinação das datas de ocorrência de cada tarefa e por haver, ao longo do tempo, inversões de prioridades impostas pela direção da empresa. Contingências operacionais também afetam a programação de serviços no curto e, eventualmente, no médio prazo. Adota-se, como hipótese do problema, o conhecimento da demanda para um horizonte de curto / médio prazo (de três a seis meses). Também se admite como determinística a duração de cada evento do problema, tais como a duração dos serviços e os tempos de deslocamento das embarcações.

A oferta é representada por embarcações (recursos) segregadas por classe. Cada embarcação é afretada segundo uma modalidade de contrato. Sendo um mercado dinâmico, os tomadores de decisão necessitam antever a tendência do mercado de afretamento, em função dos aspectos inerentes a este mercado tais como: o aquecimento da economia mundial, os fatores políticos ligados aos países produtores e exportadores de petróleo, a tendência do mercado de afretamento, de construção e de segunda mão de embarcações, entre outros. As incertezas que

permeiam este mercado fazem com que investir em novas embarcações seja uma atividade de risco. Admite-se conhecida a composição inicial da frota, os locais e os instantes em que as embarcações estão disponíveis, e as possibilidades de afretamento por classe de AHTS durante o horizonte de planejamento, caso sejam realizadas análises de composição e dimensionamento de frota.

1.3 Motivação do Tema

O tema proposto insere-se dentro da classe de problemas de roteirização e programação de veículos, tratando de um problema ainda não descrito na literatura científica. Christiansen et al. (2004) apresentam um panorama geral das publicações voltadas para o planejamento de transporte aquaviário nos níveis estratégico, tático e operacional, e discutem as especificidades inerentes a este modal que tornam particular os problemas e as contribuições nesta área.

Em geral, os problemas clássicos de roteirização e programação de veículos incorporam diversas restrições operacionais como duração dos roteiros, capacidade de carga dos veículos, janela de tempo para chegar aos clientes, entre outras. Como premissa, também se admite que a maior demanda existente (em peso ou em volume) é capaz de ser atendida pelo menor ou, por algum veículo da frota (Bodin et al., 1983), sendo possível ainda o eventual particionamento da demanda, ou o não atendimento – neste caso, mediante uma penalidade. A característica inovadora do tema desta pesquisa é a abordagem de um problema em que se faz necessário alocar múltiplas embarcações simultaneamente para realização de cada tarefa.

Do ponto de vista acadêmico, este tema é desafiante por incorporar ao problema de roteirização e programação de veículos com janela de tempo, que é um problema *NP-difícil* (Laporte; Osman, 1995), uma nova restrição operacional, que é a requisição simultânea de veículos.

Do ponto de vista prático, a programação otimizada das embarcações AHTS possibilita, no médio prazo, à empresa de petróleo distribuir a demanda de maneira balanceada, priorizando as tarefas que são essenciais para atingir suas metas econômicas e de produção. O uso racionalizado da frota é essencial, em se tratando de recursos produtivos com elevados custos fixos. Eventuais picos de demanda podem justificar o afretamento de embarcações adicionais no mercado “spot”, enquanto que períodos de baixa demanda podem servir para o agendamento de docagens, inspeções e serviços de manutenção diversos. Nestes períodos, as embarcações também podem ser empregadas no transporte de suprimentos diversos. No planejamento de longo prazo, a empresa petrolífera poderá se antecipar afretando novas embarcações, em função das descobertas de novas reservas de petróleo e das tendências observadas no mercado de construção e afretamento.

1.4 Objetivos & Metodologia

A pesquisa tem por objetivo resolver o problema de programação de frota em operações de reboque e lançamento de linhas de ancoragem para casos reais, contribuindo para o planejamento operacional dos serviços de apoio de empresas de petróleo.

De acordo com Bertrand; Fransoo (2002), a Pesquisa Operacional (“Operations Research”) como ciência, se propõe a resolver problemas na área de operações, usando como metodologia os passos identificados no ciclo da figura 1.3. O problema real, após a etapa de abstração, é convertido em um modelo conceitual, que traz consigo a especificação da extensão ou abrangência do problema real a ser resolvido, quais as variáveis relevantes deste problema e quais respostas são esperadas. 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 1.3 – Metodologia para solução de problemas em Pesquisa Operacional (adaptado de Bertrand; Fransoo, 2002).

As etapas de abstração, modelagem e resolução do modelo definem o escopo do presente trabalho. Após a descrição detalhada do problema, serão apresentadas algumas formulações matemáticas – baseadas na abordagem clássica de fluxo em rede, e orientadas à determinação do instante de início das tarefas, em uma escala de tempo discretizada. Em seguida, um conjunto métodos para solucionar o problema também será desenvolvido e aplicado.

Os métodos comumente empregados na solução de problemas de roteirização e programação de veículos possuem pelo menos duas vertentes: os baseados em estratégias otimizantes, como as técnicas “branch and bound”, “branch and cut”, “branch and price” associadas, às vezes, a técnicas de relaxação; e os modelos heurísticos construtivos, de busca local e busca estendida.

Nesta pesquisa serão apresentadas contribuições recentes de métodos construtivos para geração de solução inicial (Fischetti, et al., 2005; Achterberg; Berthold, 2005) e de métodos de busca em vizinhança (Fischetti; Lodi, 2003; Hansen et al., 2006), ambos baseados na estrutura matemática de problemas de programação linear inteira mista. No primeiro caso, a obtenção de uma solução viável se dá por meio de um processo de aproximações sucessivas, em que se busca encontrar a solução viável que esteja mais próxima de uma solução do problema relaxado quanto à integralidade das variáveis binárias. No segundo caso, a estratégia consiste em redefinir progressivamente o espaço de solução do problema, por meio da adição de restrições que limitam a busca em sub-regiões menores do que o espaço de solução do problema original. A busca se dá por meio do emprego de técnicas de enumeração implícita embutidos nos pacotes de otimização matemática. Os principais benefícios estão associados a menores tempos para obtenção de soluções de alta qualidade, quando comparados aos tempos gastos pelos pacotes de otimização convencionais.

Com relação à abordagem heurística, também será apresentada uma regra de construção probabilística, em duas versões, explorando a estrutura do problema, visando a geração de soluções de alta qualidade.

Para testar as heurísticas construtivas e as heurísticas otimizantes, foi desenvolvido um gerador de cenários em que 48 problemas de teste foram criados. Foram testados problemas com 10, 20, 30 e 40 tarefas, aproximando-se do problema em escala real vivenciado por uma empresa de petróleo, em que cerca de 50 a 60 tarefas devem ser programadas em um horizonte de aproximadamente 60 dias.

1.5 Delineamento do Texto

O presente texto está estruturado nos seguintes capítulos:

2. *Revisão Bibliográfica* – apresenta os problemas de planejamento existentes na indústria de transporte marítimo e, em seguida, classifica o problema desta pesquisa segundo a taxonomia proposta por Desrochers et al. (1990); também são revisadas as técnicas para resolução de problemas de programação inteira mista;
3. *Modelagem Matemática* – algumas formulações matemáticas serão apresentadas para representação do problema de pesquisa;
4. *Métodos de Solução* – apresenta e discute os métodos de solução desenvolvidos, incluindo a abordagem otimizante convencional, as heurísticas construtivas e os algoritmos de busca baseados na estrutura matemática do problema;
5. *Aplicação dos Métodos de Solução* – os métodos desenvolvidos são testados em um conjunto de problemas de teste; os principais resultados são apresentados e discutidos;
6. *Conclusões & Recomendações* – contém a conclusão desta pesquisa e indica possíveis extensões dos métodos desenvolvidos;
7. *Referências Bibliográficas* – lista a bibliografia consultada.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo contém a revisão da bibliografia consultada no decorrer desta pesquisa. Inicialmente, serão apresentados os principais trabalhos aplicados à indústria de transporte marítimo com ênfase na modalidade de operação denominada 'frota industrial'. Para as operações de 'linhas regulares' e 'linhas não-regulares', serão listados os principais trabalhos publicados. Em seguida, duas taxonomias para problemas de roteirização e programação de veículos serão discutidas com o intuito de enquadrar o problema desta pesquisa. Segue-se uma breve apresentação dos métodos de solução empregados em problemas de programação de veículos, bem como uma revisão dos métodos de solução para problemas de programação inteira mista.

2.1 A Indústria de Transporte Marítimo

O transporte aquaviário de matérias-primas, produtos semimanufaturados e produtos acabados tem a sua importância expressa pela significativa participação em volume (90%) e em valor (70%) dos produtos exportados mundialmente (Psaraftis, 1999). O processo de globalização levou à redefinição das cadeias de suprimento, fazendo do transporte marítimo um elo vital para garantia do abastecimento dos elementos da cadeia. A elevada participação deste modal também tem influência direta nos principais portos do mundo, que têm passado por um processo de expansão e modernização visando a compatibilização com a demanda crescente.

O segmento de transporte marítimo está organizado segundo as especificidades com que este mercado opera. Conforme Ronen (1993), as

modalidades de operação existentes podem ser classificadas em linhas regulares, linhas não-regulares e frota industrial.

A operação de linhas regulares consiste no transporte de produtos de alto valor agregado entre um conjunto de portos nomeados. A demanda existente entre cada par de portos deve ser atendida por um conjunto de rotas por meio de um serviço regular, em que o intervalo entre chegadas dos navios é aproximadamente constante em cada porto. São priorizadas a regularidade e confiabilidade das operações (nível de serviço) ao invés do custo de transporte. Períodos de baixa demanda podem levar a uma baixa ocupação dos navios, que continuam operando de forma a cumprir com as programações previstas. Picos de demanda poderão indicar a necessidade de afretamento de navios por algumas viagens. A definição de cada rota, isto é, da seqüência de portos escalados por cada navio, é uma decisão estratégica que a empresa de navegação poderá rever de tempos em tempos. As modificações, contudo, devem ser propostas sem alterar significativamente a freqüência de escala dos navios aos portos, de forma a não comprometer a imagem da empresa perante os clientes. O principal exemplo é a operação de navios porta-contêineres escalando um conjunto de portos para transporte de carga geral unitizada. Algumas referências importantes são: Rana; Vickson (1988), Perakis; Jaramillo (1991), Jaramillo; Perakis (1991), Pesenti (1995), Cho; Perakis (1996), Powell; Perakis (1997), Fagerholt (1999), Ono (2001), Lei et al. (2007) e Shintani et al. (2007).

No segmento “tramp” ou de linhas não-regulares, o armador utiliza-se de projeções de demanda para decidir quais mercados atender. Essa decisão direcionará a composição da frota por tipo de navio, tendo em vista eventuais restrições físicas dos portos em que a frota atuará. A capacidade financeira da empresa, em seguida, indicará a proporção de navios próprios e navios afretados na frota. Nestes casos, é necessário decidir a modalidade de afretamento dos navios, seja por viagem (“voyage charter”) ou por tempo (“time charter”) e por quanto tempo. Outras

questões importantes surgem, como a definição da região geográfica que empresa focará e quantas embarcações irão operar em cada região. Inerente à região, tem-se a escolha dos portos e de potenciais clientes que deverão ser priorizados. Para cada contrato de transporte que surgir, requer-se a avaliação da viabilidade de cumprimento dos níveis de serviços. Além disso, tem-se a questão de qual a melhor programação da frota em função dos contratos vigentes, dos novos contratos, dos contratos potenciais, do posicionamento atual da frota e da restrição de que determinados navios não carregam certos produtos. Também é necessário um critério para aceitar ou rejeitar um serviço e os argumentos para isso. Por último, tem-se o problema de definir a rota ótima para um navio, após o fechamento de um contrato. Os exemplos mais comuns são navios empregados no transporte de commodities (grãos, minérios, petróleo, etc.). Os seguintes trabalhos são referências em operações não-regulares: Appelgren (1969), Appelgren (1971), Fisher; Rosenwein (1989), Kim; Lee (1997), Fagerholt et al. (2000), Fagerholt (2004), Brønmo et al. (2005).

Os problemas presentes em operações com frota industrial são diversificados em função das especificidades encontradas nas várias atividades industriais. O controle da frota por parte de um grande grupo industrial se dá quando há garantia de demanda em um horizonte de médio e longo prazo. Nesta situação, além de operar com frota própria, a empresa detém o controle de embarcações afretadas em contratos “time charter” de longa duração, de forma a não correr riscos com as oscilações no mercado de fretes. Cabe à empresa alocar estes recursos da maneira mais eficaz possível, sendo predominante o objetivo de minimização do custo total da operação. Além das decisões de roteirização e programação da frota, pode ser necessária a avaliação do afretamento pontual de embarcações “spot” para atender picos de demanda. É o caso do segmento que transporta granéis líquidos e sólidos, como petróleo e derivados, minérios e fertilizantes. As características específicas de cada indústria são refletidas nas publicações existentes.

Briskin (1966) propôs um procedimento baseado na técnica de programação dinâmica para programação de navios petroleiros abastecendo um conjunto de portos, sujeitos a restrições de capacidade de armazenamento nos portos. Com base nos estoques iniciais, na taxa de consumo de petróleo e na capacidade máxima de armazenamento de cada porto, bem como na capacidade dos navios e nos tempos de viagem, o modelo proposto permite estimar os instantes dos eventos de início e término de viagem, designando cada navio a uma rota.

Perakis; Papadakis (1987) e Papadakis; Perakis (1987) propuseram uma formulação matemática detalhada do custo de transporte entre dois portos, com demanda de transporte conhecida, usando uma frota dedicada de navios. Partindo da premissa de que a frota operando com ocupação e velocidade máximas é capaz de atender toda a demanda de transporte no período desejado, então haverá uma configuração ótima de velocidade, tanto para o trecho carregado, quanto para o trecho em lastro, que minimizará o custo variável da operação. Este fato deriva da relação entre o consumo de combustível e a velocidade, o qual cresce de maneira polinomial (terceira potência) com o aumento de velocidade. Os autores avaliaram a prática do “slow steaming”, que é o ajuste da velocidade que minimiza o consumo de combustível, respeitando os prazos estipulados. Papadakis; Perakis (1989) conduziram um estudo semelhante para o caso de múltiplos portos de origem e destino.

Brown et al. (1987) deram uma abordagem analítica para o problema de transporte de petróleo do Oriente Médio abastecendo portos da América do Norte. Cada viagem era caracterizada por um porto de origem e um porto de destino; a frota era controlada por uma única empresa, sendo possível incorporar embarcações “spot” para atender picos de demanda. As viagens de retorno poderiam ser feitas por itinerários diferentes, passando pelo Canal de Suez ou pelo Cabo da Boa Esperança, tendo durações e custos diferenciados. Diversas restrições operacionais tinham que

ser observadas como não alocar determinadas cargas a navios “spot”, impedir que determinados navios fossem designados a certos portos devido a restrições de calado, entre outras. A estratégia de solução foi a elaboração de um gerador de rotas acoplado a um simulador capaz de estimar os custos das viagens. O modelo matemático para seleção de rotas foi uma variação do modelo de partição de conjuntos, o chamado “partição elástica de conjuntos”, em que algumas restrições eram relaxadas e incorporadas à função objetivo na forma de penalidades. Bausch et al. (1998) estruturaram um sistema de apoio à decisão para programação de navios incorporando este conceito de “partição elástica de conjuntos”, considerando a subdivisão dos navios em compartimentos. Bremer; Perakis (1992) e Perakis; Bremer (1992) usaram o modelo de partição de conjuntos convencional, introduzindo, contudo, no gerador de rotas, a possibilidade de sub-afretar (“out-charter”) navios da frota ao mercado “spot”.

Sherali et al. (1999) também abordaram o problema de programação de navios petroleiros abastecendo o mercado norte americano a partir do Oriente Médio. O caso estudado é rico em detalhes, com a subdivisão de cada navio em múltiplos tanques, podendo uma mesma viagem contemplar diversos produtos. O não cumprimento das janelas de tempo é permitido mediante uma penalização. O modelo matemático desenvolvido foi baseado em programação inteira mista, em que o objetivo era a minimização do custo total da operação (custos variáveis e penalidade). Foram definidas variáveis binárias para escolher se um determinado compartimento de um navio da frota carregaria um certo produto, iniciando a viagem em uma data específica, indo para um determinado destino por meio de uma das possíveis rotas. A complexidade do modelo forçou os autores a empregarem a técnica de reformulação, com a adição de desigualdades válidas (“valid inequalities”) para problemas de médio porte. Problemas de grande porte foram resolvidos com o auxílio de métodos heurísticos, explorando a estrutura do problema.

Fagerholt; Lindstad (2000) estudaram o problema de transporte de suprimentos para plataformas oceânicas, no Mar do Norte, a partir de uma base terrestre, usando navios tipo “supply”. Foi avaliado o impacto do horário de operação da base (8h00 às 16h00, ou 24hs) sobre a frota necessária e sobre os roteiros gerados. Por se tratar de uma operação em que a demanda deveria ser atendida no prazo limite de uma semana, todos os possíveis roteiros para cada embarcação candidata puderam ser gerados sem comprometer os recursos computacionais. As rotas foram posteriormente selecionadas por meio de um modelo de partição de conjuntos.

Fagerholt (2000) lidou com um problema de programação de navios, em que um conjunto de cargas deveria ser carregado em portos distintos de origem, tendo como destino um conjunto específico de portos. Associado a cada carregamento e descarregamento havia uma janela de tempo para a chegada do navio. Estas janelas são normalmente tratadas como restrições rígidas (“hard time window”), em que chegadas fora do período estabelecido configuram violação de restrições. No modelo adotado, foi introduzido o conceito de janela de tempo “soft”, em que a violação da janela de tempo é consentida até um limite, mediante uma penalização. A partir deste limite, a chegada é incompatível com a política de atendimento prevista. Este conceito foi modelado criando-se uma ‘janela de tempo interna’, em que qualquer chegada neste período não é penalizada, e uma ‘janela de tempo externa’, em que qualquer chegada antes do limite inferior da ‘janela de tempo interna’ ou após o limite superior da ‘janela de tempo interna’ é penalizada segundo uma função matemática (linear, quadrática, degrau). Isto permitiu avaliar o “trade-off” entre o nível de serviço e os custos de transporte – composto pelos custos fixos e variáveis. O problema foi resolvido por meio de um modelo de partição de conjuntos, minimizando o custo total da operação (custo fixo, variável e penalidade) usando como referência um conjunto de rotas gerado por meio de um procedimento heurístico.

Fagerholt; Christiansen (2000) trataram de um problema de programação de navios em uma operação de coleta e entrega, com janela de tempo, em que as cargas eram alocadas aos porões, isto é, os volumes de carga eram da ordem da capacidade dos porões do navio. O fato de as anteparas delimitadoras dos porões serem móveis, possibilitando o ajuste da capacidade dos porões em função do volume dos lotes embarcados, introduziu uma complexidade adicional ao problema. Este problema foi resolvido por meio de uma rotina sofisticada de geração das diversas seqüências de viagens, sendo cada roteiro otimizado segundo a técnica de programação dinâmica. O subconjunto de roteiros que atende a demanda a um mínimo custo foi escolhido por meio de um modelo de partição de conjuntos.

Xinlian; Tengfei (2000) abordaram um problema de planejamento de frota. De maneira abrangente, este estudo contemplou decisões sobre afretamento e dispensa de navios da frota bem como a forma ótima de utilização dos navios, alocando-os às rotas. Trata-se de uma operação de transporte na qual um conjunto de portos ofertantes necessita remeter carga para um conjunto de portos receptores, formando um conjunto de rotas, em que cada rota contempla apenas um porto de origem e um porto de destino. Existem diversos tipos de navios que podem, eventualmente, não fazer parte da frota no início do período de planejamento, mas que poderão ser incorporados ao longo do tempo. O horizonte de planejamento é composto de diversos períodos ou anos. São decisões do modelo: 1) o plano ótimo de alocação da frota às rotas em cada ano; 2) quais tipos de navio e quantos deverão ser selecionados e incorporados à frota anualmente, caso haja um excedente na demanda; 3) quais tipos de navio e quantos deverão ser selecionados para serem fundeados (ou dispensados da frota) e que tonelagem ficará ociosa em cada ano. Para resolução, o problema foi decomposto em uma série de decisões consecutivas, realizadas no início de cada ano. Empregou-se a técnica de programação dinâmica para realizar o planejamento da frota, quanto à incorporação e dispensa de navios, tendo como

estágio cada ano do horizonte de planejamento. Em cada estágio, o estado do sistema é definido pela composição da frota (quantidade de navios dos diversos tipos utilizados pela empresa).

Cho; Perakis (2001) deram uma nova abordagem ao problema estudado por Ronen (1986), em que uma frota de navios deve ser programada para, a partir de um único porto, abastecer um mercado regional de grãos sólidos. O modelo matemático proposto foi uma generalização do problema capacitado de localização de instalações (“capacitated facility location problem”).

Christiansen; Fagerholt (2002) trataram de um problema de programação de navios em que múltiplas cargas deveriam ser carregadas em um conjunto de portos de origem e descarregadas em um conjunto de portos de destino. Além da janela de tempo prevista para a chegada dos navios nos diversos portos, o problema apresentava restrições de operação dos portos, em que alguns operavam apenas 8 horas por dia e fechavam aos finais de semana. Daí ser necessária uma programação robusta de forma que as embarcações não chegassem aos portos em horários que oferecessem riscos de serem retidas à noite ou por todo um final de semana. A estratégia foi determinar ‘janelas de risco’, isto é, janelas que causariam a retenção das embarcações, e penalizar as chegadas nestas janelas. Isto foi modelado com o auxílio de um gerador de rotas para construção de todas as possíveis seqüências de viagens. Cada rota gerada foi otimizada com a aplicação de um algoritmo de programação dinâmica. Em seguida, as rotas mais econômicas foram escolhidas por meio de um modelo de partição de conjuntos que contemplava a decisão de afretar embarcações “spot” para o atendimento de toda a demanda.

2.2 Introdução à Taxonomia do Problema de Pesquisa

Os problemas de roteirização e programação de veículos (PRPVs), englobando os diversos modais de transporte, têm recebido atenção especial do setor produtivo bem como do meio acadêmico nas últimas décadas. A literatura científica nesta área é significativamente extensa. Laporte; Osman (1995) revisaram 500 artigos com as principais contribuições teóricas e aplicações. Revisões recentes (Bräysy; Gendreau, 2005a, 2005b) têm demonstrado que a produção nesta área tem sido contínua e crescente.

A diversidade de aplicações práticas dos PRPVs requer uma taxonomia que permita enquadrar os diversos problemas e avaliar as contribuições reais que são feitas nesta área. Bodin et al. (1983) propuseram uma primeira sistemática de classificação para os problemas de roteirização, incorporando os aspectos gerais encontrados nestas operações, mostrada na tabela 2.1.

Com o objetivo de justificar a contribuição da presente pesquisa, procurou-se uma outra classificação para os PRPVs que pudesse evidenciar a principal restrição operacional do problema em questão, que é a requisição simultânea de embarcações na execução de uma tarefa.

A taxonomia encontrada foi a de Desrochers et al. (1990), reapresentada em Desrochers et al. (1999). Esta taxonomia, que será, em seguida apresentada, requer o detalhamento dos aspectos: 1) endereços (loais que ofertam ou demandam produtos ou serviços; 2) veículos; 3) características do problema e; 4) objetivos.

Tabela 2.1 – Características dos Problemas de Roteirização (Bodin et al., 1983)

Característica	Aspectos
Tamanho da Frota Disponível	Um Veículo
	Diversos Veículos
Tipo da Frota Disponível	Homogênea
	Heterogênea
Número de Garagens	Uma Garagem
	Várias
Natureza da Demanda	Determinística
	Estocástica
	Atendimento Parcial é Permitido
Rede Associada	Orientada
	Não Orientada
	Mista
Localização da Demanda	Nos Nós da Rede
	Nos Arcos da Rede
	Em Arcos e Nós da Rede
Restrições de Capacidade dos Veículos	Imposta e Única
	Imposta e Múltipla
	Não Imposta
Tempo Máximo de Viagem	Imposto e Único (para todas as rotas)
	Imposto e Diversificado (diferente para rotas diferentes)
	Não Imposto
Tipo de Operação	Coleta
	Entrega
	Coleta e Entrega
Custos	Custos Variáveis ou de Roteirização
	Custos Fixos
	Custo do Transporte Alternativo para a Demanda Não Atendida
Objetivos	Minimizar o Custo Total de Roteirização
	Minimizar a Soma de Custos Fixos e Variáveis
	Minimizar a Frota Afretada
	Minimizar o Número de Veículos Necessários
	Minimizar a Função de Utilidade Baseada em Nível de Serviço e Prioridade dos Clientes
	Balanceamento das Rotas

Elementos da taxonomia de Desrochers et al. (1990):

1. *Endereços* – os locais de oferta e demanda são caracterizados, com base nos parâmetros:

Número de depósitos – podendo ser uma ou várias bases de origem (loais de oferta);

Tipo de demanda – esta pode estar localizada nos nós ou em arcos de uma rede (ou em ambos); o problema pode ser exclusivo de coleta, entrega ou misto; por último, a natureza da demanda pode ser determinística ou estocástica;

Restrições de programação dos endereços – cada cliente da rede pode, eventualmente, impor janela de tempo para o início do atendimento. Esta janela, se houver, poderá ser específica ou geral a todos os clientes;

Restrições de seleção dos endereços – em determinados problemas, nem todos os clientes necessariamente têm que ser atendidos. Este tipo de restrição permite estabelecer esta não obrigatoriedade, impondo, eventualmente, que apenas um subconjunto de clientes deva ser atendido. Outras opções são a de atender pelo menos um cliente em cada subconjunto de determinadas partições;

2. *Veículos* – as características da frota a ser empregada são evidenciadas pelos parâmetros:

Número de veículos – especifica a quantidade e se todos os veículos devem ser empregados;

Capacidade dos veículos – indica se os veículos têm limitação de capacidade de carga, se as capacidades são iguais ou diferenciadas por veículo;

Restrições de tipo de produto – informam se os veículos possuem limitação quanto ao tipo de produto que podem transportar, se há compartimentos de carga dedicados ou intercambiáveis;

Restrições de programação de veículos – impõem as janelas contratuais de operação dos veículos, isto é, os períodos em que os veículos estão disponíveis para a operação;

Restrições de duração das rotas – impõem o limite de duração de cada roteiro dos veículos;

3. *Características do problema* – este grupo especifica aspectos principais do problema de roteirização e programação:

Tipo de rede – parâmetro que informa se na rede é válida a propriedade de desigualdade triangular para cálculos das distâncias e dos custos, ou não. Também nesse aspecto é necessário informar se a rede é orientada, não-orientada ou mista;

Tipo de estratégia – diz respeito à possibilidade de particionamento da demanda e, se este eventual particionamento é feito *a priori*, ou durante a elaboração dos roteiros. O tipo de estratégia também é definido pela obrigatoriedade de se buscar carga de retorno, bem como se a operação é de carga fracionada ou carga lotação⁶. Também é necessário definir se haverá no máximo uma rota por veículo ou, se um veículo pode ser alocado a mais de uma rota. Por último, especifica-se a estratégia pela imposição de uma rota começar e terminar na mesma base, ou se o retorno a qualquer outra base é permitido.

Restrição entre endereços – ocorre quando há precedência entre endereços (ou clientes), quando há restrição de uma determinada base em atender algum cliente, ou quando há restrição quanto ao atendimento de dois ou mais clientes específicos em uma mesma rota;

Restrição endereços-veículos – ocorre quando algum veículo não pode ser atendido em uma determinada base, ou quando algum cliente não pode receber algum veículo;

Restrição entre veículos – ocorre quando se impõe sincronia ou, simultaneidade, na operação com mais de um veículo;

4. *Objetivos* – em geral, são expressos pela minimização de uma função de mérito ou pela minimização do máximo valor que uma função de mérito pode assumir. A função de mérito pode incorporar as parcelas de: **custo variável**, que é função da

⁶ Operações com carga fracionada ocorrem tipicamente em problemas de distribuição física, em que a demanda de cada cliente lota uma fração do veículo. Nestes problemas, os veículos devem ser roteirizados, de forma a atender a demanda de vários clientes, respeitando a capacidade dos veículos. Carga lotação é o termo empregado para operações em que os veículos têm sua capacidade integral aproveitada por um único cliente.

distância ou da duração das rotas; **custo fixo**, que depende do tamanho da frota utilizada; **penalidade** associada ao uso de um veículo; **custo de atendimento** dos cliente e; **penalidade** associada ao nível de serviço praticado.

A tabela 2.2 apresenta a taxonomia de Desrochers et al. (1990), indicando as características do problema desta pesquisa.

Tabela 2.2 - Taxonomia de Desrochers et al. (1990)

ENDEREÇOS		Problema desta Pesquisa
Número de depósitos	Um depósito; múltiplos depósitos	<i>um depósito⁷</i>
Tipo de demanda	Localizada em nós; arcos; ambos; orientada a tarefas	<i>nós</i>
	Coleta; entrega; misto	<i>não se aplica</i>
	Natureza: determinística; estocástica	<i>determinística</i>
Restrições de programação dos endereços	Sem restrições; janela única; janela específica por cliente	<i>janela de tempo específica para cada cliente</i>
Restrições de seleção dos endereços	Atendimento integral, parcial, parcial por subconjunto de clientes	<i>todos os clientes devem ser atendidos</i>
VEÍCULOS		Problema desta Pesquisa
Número de veículos	Quantidade	<i>valor informado</i>
	Todos os veículos devem ser empregados?	<i>nem todos precisam ser empregados</i>
Capacidade dos veículos	Não há; igual para todos; distinto por veículo	<i>não há</i>
Restrições de tipo de produto	Sem compartimentos; compartimentos dedicados; compartimentos intercambiáveis	<i>não se aplica</i>
Restrições de programação de veículos	Não há restrição; todos os veículos possuem a mesma janela operacional; cada veículo possui uma janela de operação específica	<i>Cada veículo possui um contrato de afretamento específico</i>
Restrições de duração das rotas	Não há restrição; todos os veículos possuem a mesma autonomia; cada veículo possui uma autonomia específica	<i>cada veículo possui uma autonomia específica</i>

⁷ Embora haja uma única base, as embarcações não precisam estar na base no início do horizonte de programação, e nem necessitam voltar para a base após o término da execução de suas tarefas.

Tabela 2.2 - Taxonomia de Desrochers et al. (1990) – Continuação

CARACTERÍSTICAS DO PROBLEMA		Problema desta Pesquisa
Tipo de rede	Vale a desigualdade triangular; custos informados	<i>custos de transporte informados</i>
	Rede orientada; não-orientada; mista	<i>rede orientada</i>
Tipo de estratégia	Particionamento da demanda (não; sim <i>a priori</i> , sim <i>a posteiori</i>)	<i>a demanda não é particionada</i>
	Há carga de retorno? É obrigatório carga lotação?	<i>não se aplica</i>
	Uma ou múltiplas rotas por veículo	<i>cada veículo executa uma rota</i>
	Retorna para a mesma base ou para outra	<i>não é necessário o retorno para a base</i>
Restrição entre endereços	Há precedência entre endereços? Há restrição base-endereço? Há restrição entre endereços?	<i>há restrição de precedência entre tarefas</i>
Restrição endereços-veículos	Há restrição base-veículo? Há restrição endereço-veículo?	<i>há imposição de tipos de veículos para realizar uma tarefa</i>
Restrição entre veículos	São necessários múltiplos veículos para realizar uma operação?	<i>as tarefas requerem múltiplas embarcações simultaneamente</i>
OBJETIVOS		Problema desta Pesquisa
Operador	Minimizar função; minimizar o máximo valor que uma função pode assumir	<i>minimizar função</i>
Funções	Custo da roteirização (variáveis), custo dos veículos (fixo), penalidade associada aos veículos, custo de atendimento, penalidade associada a nível de serviço	<i>custos variáveis e penalidade associada a nível de serviço</i>

A característica particular do problema desta pesquisa ficou, portanto, destacada no item “restrições entre veículos”, em que a sincronização é um requisito da operação de transporte.

A pesquisa bibliográfica indicou que as ocorrências de sincronização entre veículos estão ligadas ao problema de designação de locomotivas. Neste problema existem diversos trechos de viagens compondo o quadro de itinerários de uma empresa ferroviária que devem ser atendidos pelas composições ferroviárias (locomotivas e vagões). Alguns trechos, por apresentarem maior declividade ou por receberem composições de maior porte, requisitam múltiplas locomotivas operando simultaneamente, de forma a permitir a passagem das composições dentro de

padrões mínimos de segurança, mantendo as velocidades médias desejadas (Cordeau et al., 1998; Liu, 2003). A principal diferença em relação ao problema de roteirização e programação de veículos é que as locomotivas devem ser designadas às rotas pré-estabelecidas ao invés de roteirizadas.

Uma aplicação recente (Francis et al., 2007) foi desenvolvida para a definição de rotas em uma operação de reposicionamento de veículos. Neste problema, existem “pools” de carretas espalhadas geograficamente nas diversas bases de uma empresa e um conjunto de requisições de transporte para que as carretas vazias sejam posicionadas nos clientes, isto é, nos embarcadores. Enquanto que algumas requisições permitem que as carretas sejam deslocadas a partir de qualquer base, outras são específicas por base, por estas terem estoque de veículos específicos. A roteirização dos múltiplos recursos simultâneos se dá pela associação do cavalo mecânico à carreta para atender à demanda de transporte. Apesar de ser caracterizado como um problema de roteirização de veículos, este problema assemelha-se mais aos problemas de designação, como no caso das locomotivas.

2.3 Métodos de Solução Empregados em Problemas de Roteirização e Programação de Veículos

O problema de roteirização e programação de veículos com janelas de tempo é uma extensão do problema clássico de roteirização de veículos, ao impor que cada cliente seja atendido dentro de uma janela de tempo. Nestes problemas, cada veículo inicia e termina o roteiro na base, e as restrições de duração da rota e capacidade de carga devem ser respeitadas. A duração de cada rota é composta dos tempos de viagem e do tempo gasto em cada um dos clientes. Bodin et al. (1983) apresenta a formulação clássica destes problemas, baseada em um problema de fluxo em rede. É objetivo do problema de roteirização e programação de veículos a minimização do

número de rotas, que reflete a quantidade de veículos necessários, bem como a distância total percorrida. Alguns autores focam a minimização combinada de ambos objetivos, enquanto outros trabalhos enfatizam apenas um aspecto.

Cordeau et al. (2005) mostram que o método clássico “branch and bound” é incapaz de obter a solução exata para os problemas de roteirização e programação de veículos, mesmo em se tratando de problemas de pequeno porte. A justificativa para isso é a baixa qualidade do limitante gerado pela versão relaxada do problema. As abordagens exatas que apresentam maior sucesso, segundo estes autores, são as técnicas de relaxação lagrangeana, o método de geração de colunas e a estratégia “branch and cut”.

Bräysy; Gendreau (2005a, 2005b) apresentam um resumo das contribuições recentes das abordagens heurísticas para os PRPVs. Inicialmente, são listadas as heurísticas construtivas, que consistem em regras empregadas para a construção de uma solução viável para o problema. Cada cliente é adicionado à rota mediante um critério de escolha do “próximo” cliente e da escolha da posição de inserção. Este processo é realizado enquanto a adição mantiver a viabilidade da solução gerada e até que todos os clientes sejam alocados. Alguns trabalhos clássicos são a heurística do roteiro gigante (Golden et al., 1984), a adaptação do método das economias de Clarke; Wrigth (1964) com a inserção de viabilidade temporal e a heurística de inserção I1 (Solomon, 1987), entre outros. Bräysy; Gendreau (2005a) apresentam uma listagem abrangente e atualizada das principais heurísticas.

As heurísticas de busca consistem em estratégias de melhoria de uma solução inicial, sem a garantia de geração da solução ótima. O emprego de um método de busca local requer a prévia definição de: uma solução inicial viável que será explorada; da forma de geração da vizinhança; do critério de aceitação de uma solução gerada; e de um critério de parada. A geração da vizinhança ocorre em

função do mecanismo empregado para criar novas soluções a partir da solução corrente como, por exemplo, a troca da posição de clientes e a substituição de arcos, entre outras. Se a solução na vizinhança da solução corrente é melhor, então esta se torna a solução corrente, substituindo a anterior. Os critérios de aceitação comumente empregados são: ‘escolhe-o-primeiro’ (“first-accept”), em que a primeira solução gerada que for melhor que a corrente é escolhida ou, ‘escolhe-a-melhor’ (“best-accept”) de todas as soluções na vizinhança da solução corrente. Caso não haja uma solução melhor, ter-se-á chegado a uma solução ótima local e o algoritmo encerra.

O método clássico 2-Opt (Lin, 1965) adaptado para os PRPVs, tem sido aplicado, explorando a vizinhança dentro de uma mesma rota ou trocando clientes entre rotas. Diversos mecanismos de geração de vizinhança também têm sido desenvolvidos e aplicados com sucesso aos PRPVs, entre eles os operadores “2-Opt*” (Potvin; Rousseau, 1995), “relocate, exchange, cross” (Savelsbergh, 1992), “CROSS” (Taillard et al., 1997) “GENI” (Gendreau et al., 1992) e “cyclic transfer” (Thompson; Orlin, 1993), entre outros. Bräysy; Gendreau (2005a) apresentam uma listagem completa e atualizada das principais heurísticas de busca local.

Em geral, o processo de busca local pode levar a soluções sub-ótimas, se nenhuma solução na vizinhança da solução corrente apresentar melhoria. Os métodos de busca estendida admitem uma solução pior que a solução corrente, com o objetivo de estender a busca para outras regiões do espaço de solução. Estes métodos, denominados de metaheurísticos, têm tido aceitação na comunidade acadêmica na solução de problemas combinatórios, pela possibilidade de alcançar soluções de qualidade, embora a um esforço computacional maior. Os métodos mais comuns são a busca tabu, os algoritmos genéticos e o “simulated annealing”.

A principal idéia da busca tabu (Glover; Laguna, 1999) é forçar a exploração de novas áreas do espaço de solução, por meio de uma memória que armazena soluções que foram examinadas recentemente, impedindo-as de serem re-visitadas (movimentos tabus). Quando a estratégia de intensificação não conseguir gerar mais soluções de boa qualidade, aplica-se a estratégia de diversificação, que consiste em forçar movimentos ou soluções ainda não explorados, mesmo que penalize a função objetivo. Isto permitiria explorar uma outra região do espaço de solução.

O algoritmo genético foi introduzido por Holland (1975). É um método que parte de um conjunto de n soluções viáveis de um problema e, em cada iteração, gera um novo conjunto de m ($m \geq n$) novas soluções mediante a aplicação de operadores genéticos. Estas novas soluções, ou indivíduos, são avaliados segundo uma função de “fitness” (função de mérito), e apenas os n melhores indivíduos do total de $m+n$ indivíduos sobrevivem. É o conceito de seleção natural, onde apenas os indivíduos mais aptos sobrevivem, formando uma nova geração. Este processo é repetido até que o critério de parada seja atendido. Em geral, este critério é representado pelo número de iterações, ou pelo tempo total de processamento, ou pelo número de iterações sem melhoria. Os operadores genéticos indicam a forma de geração de novos indivíduos. Basicamente dois operadores são empregados: 1) cruzamento – quando há um cruzamento entre dois indivíduos para gerar um novo elemento. Para isso, são escolhidos, de maneira aleatória, dois indivíduos de uma geração, os quais cederão parte de seu material genético na formação do novo indivíduo. Isto significa que o elemento gerado será constituído por genes de ambos os pais; 2) mutação – ocorre quando um indivíduo tem parte de seu cromossomo alterado, tornando-o um novo indivíduo. A intensificação da busca pode ser entendida como o processo de geração de novos indivíduos segundo o processo natural de “crossover”. Numa nova geração, contudo, um descendente poderá ter uma modificação aleatória (mutação) dentro do seu próprio cromossomo não resultante de sua herança genética. Dentro do conceito de busca estendida, este é um mecanismo que permite a geração de uma

solução totalmente nova, de forma a escapar de regiões subótimas sendo, portanto, o mecanismo de diversificação do algoritmo genético.

O “simulated annealing” – SA (Kirkpatrick et al., 1983) é uma metaheurística inspirada no processo físico de resfriamento de materiais em um processo de tratamento térmico. A busca inicia-se a partir de uma solução inicial. Uma solução na vizinhança desta será aceita se for melhor que a solução de referência (a melhor solução conhecida). Se for pior, ela será aceita mediante uma probabilidade de referência, que é proporcional à diferença entre o valor da função objetivo na solução da vizinhança e o valor da função objetivo na solução de referência. Esta diferença é normalizada por um parâmetro, que é atualizado durante a busca. A diferença entre as soluções representa a variação de temperatura entre dois estágios de resfriamento; o parâmetro de normalização é a temperatura inicial do processo de resfriamento, e a atualização do parâmetro é dada pela taxa de resfriamento do material.

Pardalos; Resende (2002) compilaram as metaheurísticas mais conhecidas e Bräysy; Gendreau (2005b) mostram quais implementações têm se destacado na solução do PRPV.

2.4 Métodos de Solução Empregados em Problemas de Programação Linear Inteira Mista

Um problema de programação linear inteira mista (MIP - “mixed integer program”) é um problema de programação matemática envolvendo variáveis inteiras, variáveis reais, função objetivo e restrições lineares. Qualquer MIP pode ser escrito como:

$$(MIP) \quad Z(X) = \min_{(x,y)} \{cx + fy : (x, y) \in X\} \quad (2.1)$$

O conjunto X é chamado de conjunto de soluções viáveis e é descrito por m restrições lineares, restrições de não-negatividade sobre x e y , e restrições de integralidade sobre os valores de y . A notação matricial para X é:

$$X = \{(x, y) \in R_+^n \times Z_+^p : Ax + By \geq b\} \quad (2.2)$$

onde:

$Z(X)$ é o valor ótimo da função objetivo;

x e y são, respectivamente, o vetor coluna n -dimensional de variáveis reais não-negativas, e o vetor coluna p -dimensional de variáveis inteiras não-negativas;

$c \in R^n$ e $f \in R^p$ são vetores linha dos coeficientes da função objetivo;

$b \in R^m$ é o vetor coluna dos termos independentes das m restrições;

A e B são as matrizes dos coeficientes reais das variáveis x e y nas restrições, de dimensões $(m \times n)$ e $(m \times p)$, respectivamente.

O problema de programação linear relaxado (problema LR) associado ao MIP, e o seu valor ótimo $Z(P_X)$, são definidos pelo conjunto P_X , em que as variáveis y são relaxadas quanto à integralidade:

$$P_X = \{(x, y) \in R_+^n \times R_+^p : Ax + By \geq b\} \quad (2.3)$$

Esta seção irá destacar os principais métodos usados para resolver problemas de programação inteira mista (MIP), conforme a revisão de Pochet; Wolsey (2006).

2.4.1 Branch and Bound

Um dos métodos mais empregados para a resolução de problemas de programação inteira mista é o “branch and bound” (B&B). No B&B, diversos

subproblemas P_V serão gerados, cujos valores da função objetivo serão dados por:

$Z(P_V) = \min_{(x,y)} \{cx + fy : (x,y) \in V\}$. Se o subproblema P_V gerado for inviável, por convenção $Z(P_V) = +\infty$.

A estratégia do método B&B baseia-se nos fatos: 1) a solução ótima do problema relaxado é um limitante inferior para a solução ótima do problema original ($Z(P_X) \leq Z(X)$); 2) qualquer solução viável (x,y) do problema original fornecerá um limitante superior $Z = cx + fy$ para a solução do problema ($Z(X) \leq Z$).

Os princípios de funcionamento do “Branch and Bound” são:

1) O limitante inferior da solução ótima do problema $Z(X)$ é fornecido pelo valor da solução ótima do problema relaxado $Z(P_X)$ ⁸. Seja (x^*, y^*) a solução ótima do problema relaxado LR .

2) Se $y^* \in Z^p$, então a solução é viável quanto à integralidade e $(x^*, y^*) \in X$ fornecerá um limitante superior para a solução. Por ser, ao mesmo tempo, um limitante inferior e superior, a solução encontrada é, comprovadamente, a ótima ($cx^* + fy^* = Z(P_X) \leq Z(X) \leq \bar{Z} = cx^* + fy^*$).

3) Ramificação. Caso contrário, $y^* \notin Z^p$ e a solução encontrada (x^*, y^*) não é viável para o problema original. Esta inviabilidade é contornada pela adição de restrições lineares ao problema. Seja $y_j, j \in \{1, \dots, p\}$ uma variável com valor não inteiro na solução (x^*, y^*) de LR (variável de ramificação ou “branching variable”). O conjunto P_X será substituído pela união de dois conjuntos disjuntos P_X^0 e P_X^1 :

$$P_X^0 = P_X \cap \{(x, y) \in R_+^n \times R_+^p : y_j \leq [y_j^*]\} \text{ e } P_X^1 = P_X \cap \{(x, y) \in R_+^n \times R_+^p : y_j \geq \langle y_j^* \rangle\}$$
⁹

4) O próximo passo procura pela melhor solução inteira em alguma das formulações contidas na lista $L = \{P_X^0, P_X^1\}$.

⁸ O limitante pode também ser obtido por meio da técnica de relaxação lagrangeana.

⁹ $[x]$ representa o maior inteiro menor ou igual a x ; $\langle x \rangle$ represente o menor inteiro maior ou igual a x .

A iteração principal do método B&B consiste em: dada uma lista L de formulações, e o valor da melhor solução inteira já encontrada $\bar{Z}$ (caso ainda não tenha sido encontrada nenhuma solução inteira, $\bar{Z} = +\infty$), selecionar um problema P_V da lista e resolver o problema linear, obtendo a solução (x^V, y^V) cujo valor é $Z(P_V)$, que será um limitante inferior para a melhor solução inteira do problema P_V . Alguns casos podem ocorrer:

Eliminação

- a. Se $Z(P_V) \geq \bar{Z}$, então se caracterizaria uma situação em que a solução ótima do problema P_V é pior do que o melhor limitante superior já encontrado. Ficaria descartada a necessidade de explorar soluções inteiras em P_V e este subproblema seria retirado da lista L . A eliminação seria por exceder o limitante (“pruning by bound”).
- b. Um caso particular é quando o problema P_V não tem solução viável. Neste caso, $Z(P_V) = +\infty \geq \bar{Z}$ e a eliminação de P_V da lista L se daria pela inviabilidade (“pruning by infeasibility”).
- c. Se $Z(P_V) < \bar{Z}$ e $y^V \in Z^p$, então uma solução viável (x^V, y^V) para o problema original foi encontrada e o limitante superior será atualizado para $\bar{Z} = Z(P_V)$. Não haveria necessidade de incluir o problema P_V na lista L , devido à integralidade da solução encontrada (“pruning by integrality”). Além disso, todos os problemas P_W em L , tal que $Z(P_W) \geq \bar{Z}$ poderão ser removidos da lista.
- d. Se $Z(P_V) < \bar{Z}$ e $y^V \notin Z^p$, então na solução ótima do problema de programação linear P_V há pelo menos uma variável y_j com valor não inteiro; o problema P_V deverá ser removido da lista L , dando lugar a dois subproblemas P_V^0 e P_V^1 , gerados conforme descrito no passo 3.

Término – O algoritmo encerra quando a lista L fica vazia.

De maneira esquemática, o método B&B é apresentado na figura 2.1.

Algoritmo Branch and Bound

1. Inicialização

$$L = \{P_X\}$$

$$\bar{Z} = +\infty$$

Considerar que LR é limitado ($Z(P_X) > -\infty$)

2. Término

se $L = \emptyset$ **então**

se $\bar{Z} = +\infty$ **então** $X = \emptyset$ (problema inviável)

se $\bar{Z} < +\infty$ **então** a solução $(x, y) \in X$ com $\bar{Z} = cx + fy$ é ótima

fim

3. Seleção de um nó e solução do problema

Selecionar $P_V \in L$ e fazer $L = L \setminus \{P_V\}$

Achar a solução ótima (x^V, y^V) e o valor $Z(P_V)$ do problema P_V

4. Eliminação

se $Z(P_V) \geq \bar{Z}$ **então** ir para o passo 2 (P_V é inviável ou dominado pela melhor solução inteira já encontrada)

se $Z(P_V) < \bar{Z}$ **então**

se $y^V \in Z^p$ **então** (a solução é integral)

atualizar o limitante superior fazendo $\bar{Z} = Z(P_V)$

atualizar a lista L removendo os problemas dominados

(para cada $P_W \in L$: **se** $Z(P_W) \geq \bar{Z}$, **então** $L = L \setminus \{P_W\}$)

ir para o passo 2

5. Ramificação

(ocorre apenas quando P_V não foi eliminado: $Z(P_V) < \bar{Z}$ e $y^V \notin Z^p$)

Selecionar j tal que $y_j^V \notin Z$

Atualizar a lista L :

$$L = L \cup \{P_V^0, P_V^1\} \text{ onde:}$$

$$P_V^0 = P_V \cap \{(x, y) \in R_+^n \times R_+^p : y_j \leq [y_j^V]\}$$

$$P_V^1 = P_V \cap \{(x, y) \in R_+^n \times R_+^p : y_j \geq [y_j^V] + 1\}$$

Ir para o passo 2

Figura 2.1 – Pseudo código do algoritmo “Branch and Bound” (adaptado de Pochet; Wolsey, 2006)

2.4.2 Reformulação

O método “branch-and-bound” (B&B) descreve uma estratégia geral de busca da solução ótima que, na prática, pode ser comprometida pelo número de problemas

gerados, que cresce exponencialmente com o número de variáveis inteiras. Não só o porte do problema é um fator determinante do número de nós da árvore B&B. A formulação matemática, isto é, as variáveis de decisão e a estrutura das restrições também têm influência direta no desempenho dos métodos B&B.

A estratégia de reformulação visa introduzir na estrutura matemática do problema novas restrições lineares, chamadas de desigualdades válidas (VI – “valid inequalities”) que não modificam o espaço de solução do problema. Objetiva-se, com isso, obter-se uma formulação mais restrita de forma a melhorar os limitantes inferiores providos pelos problemas relaxados resolvidos em cada nó.

O caso mais abrangente é quando todos os pontos extremos que compõem o espaço de solução de um problema são envolvidos por um conjunto de desigualdades, formando um poliedro convexo ($conv(X)$ – poliedro do espaço de solução de um problema X), configurando-se na formulação mais restrita possível (“tightest valid formulation”). A solução ótima do problema relaxado LR necessariamente seria ótima para o problema MIP. Embora conceitualmente interessante, o método esbarra na dificuldade em conseguir definir o poliedro $conv(X)$. E, para a maioria das instâncias de problemas MIP, o número de desigualdades é excessivamente elevado (Pochet; Wolsey, 2006).

Uma desigualdade válida (VI) para o conjunto viável X de MIP do tipo $X = \{(x, y) \in R_+^n \times Z_+^p : Ax + By \geq b\}$ é uma restrição, ou desigualdade, $\alpha x + \beta y \geq \gamma$ ($\alpha \in R^n, \beta \in R^p, \gamma \in R$) que é satisfeita por todos os pontos em X : $\alpha x^* + \beta y^* \geq \gamma$ para todo $(x^*, y^*) \in X$. São de particular interesse as desigualdades válidas que contribuem para a definição do poliedro convexo $conv(X)$.

Devido à inviabilidade de incorporar no B&B todo o conjunto de desigualdades válidas que formam o poliedro convexo, é comum utilizar-se de uma família C de desigualdades válidas, que são adicionadas ao modelo de maneira progressiva, representando cortes com potencial de acelerar a obtenção de bons limitantes. Tais desigualdades são definidas como:

$$C = \{\alpha^j x + \beta^j y \geq \gamma^j \text{ para todo } j = 1, \dots, |C|\}. \quad (2.4)$$

A reformulação de X em função de C passa ser:

$$\tilde{X} = \left\{ (x, y) \in R_+^n \times Z_+^p : \begin{array}{l} Ax + By \geq b \\ \alpha^j x + \beta^j y \geq \gamma^j \text{ para todo } j = 1, \dots, |C| \end{array} \right\} \quad (2.5)$$

Dessa forma, a região viável X reformulada ($\tilde{X}$) passa a ser: $\tilde{X} = X \cap C \subseteq X$.

Uma das estratégias, conhecida como *reformulação à priori*, utiliza $\tilde{X}$ na estrutura do algoritmo B&B, sendo eficiente apenas em casos em que o número de desigualdades adicionadas $|C|$ é pequeno.

Os pacotes de otimização matemática disponíveis no mercado trabalham com algumas famílias de desigualdades. Ao invés de serem específicas para um problema, estas desigualdades são aplicáveis à maior parte dos problemas de programação inteira, pois abordam aspectos gerais como, por exemplo, a proposição de cortes para eliminação de soluções não-inteiras.

Algumas das famílias de desigualdades válidas serão revisadas, em particular, as que acompanham o software de otimização empregado nesta pesquisa – CPLEX 10.0. O conjunto de resultados a seguir apresentados foi extraído e adaptado de Wolsey (1998).

2.4.2.1 Desigualdade Válida de Chvátal-Gomory (Gomory's Fractional Cutting Plane Algorithm)

Seja $P = \{x \in R_+^n : Ax \leq b\}$ um conjunto de soluções viáveis de um problema de programação linear e $X = P \cap Z^n$ o conjunto de soluções que atendam às restrições $Ax \leq b$ e x inteiro, não-negativo. A é uma matriz $m \times n$, com colunas $\{a_1, a_2, \dots, a_n\}$, e seja $u \in R_+^m$. Então:

- (i) a desigualdade $\sum_{j=1}^n ua_j x_j \leq ub$ é válida para P , já que $u \geq 0$ e $\sum_{j=1}^n a_j x_j \leq b$;
- (ii) a desigualdade¹⁰ $\sum_{j=1}^n [ua_j] x_j \leq ub$ é válida para P pois $x \geq 0$;
- (iii) a desigualdade $\sum_{j=1}^n [ua_j] x_j \leq [ub]$ também será válida para X pois x é inteiro e

$$\sum_{j=1}^n [ua_j] x_j \text{ é inteiro.}$$

A desigualdade de Chvátal-Gomory pode ser aplicada toda vez que a resolução do problema linear em um nó da árvore “branch and bound” indicar solução fracionária, isto é, solução em que pelo menos uma variável inteira no problema original apresentar valor não-inteiro na solução do problema linear relaxado.

Seja o problema de programação inteira $\max\{cx : Ax = b, x \in Z_+^n\}$. Supor que a forma canônica correspondente à solução básica ótima do problema linear seja escrita da seguinte forma:

¹⁰ O operador $[x]$ devolve o maior inteiro menor ou igual a x .

$$\begin{aligned}
& \max \bar{a}_{00} + \sum_{j \in NB} \bar{a}_{0j} x_j \\
& x_{Bu} + \sum_{j \in NB} \bar{a}_{uj} x_j = \bar{a}_{u0} \quad u=1, \dots, m \\
& x \geq 0
\end{aligned} \tag{2.6}$$

Nesta formulação, NB é o conjunto de variáveis não-básicas, $\bar{a}_{0j} \leq 0$, $j \in NB$ e $\bar{a}_{u0} \geq 0$ para $u=1, \dots, m$. Se, na solução básica ótima ($x_j = 0$, $j \in NB$, $x_{Bu} = \bar{a}_{u0}$, $u=1, \dots, m$) do problema de programação linear relaxado, houver algum $x_{Bu}(\bar{a}_{u0})$ não inteiro, então:

$$\begin{aligned}
& x_{Bu} + \sum_{j \in NB} [\bar{a}_{uj}] x_j \leq \bar{a}_{u0} \quad \text{e, pela desigualdade de Chvátal-Gomory,} \\
& x_{Bu} + \sum_{j \in NB} [\bar{a}_{uj}] x_j \leq [\bar{a}_{u0}]
\end{aligned} \tag{2.7}$$

Como pelas equações da forma canônica $x_{Bu} + \sum_{j \in NB} \bar{a}_{uj} x_j = \bar{a}_{u0}$, $u=1, \dots, m$, resulta então:

$$\begin{aligned}
& \sum_{j \in NB} (\bar{a}_{uj} - [\bar{a}_{uj}]) x_j \geq \bar{a}_{u0} - [\bar{a}_{u0}] \text{ ou} \\
& \sum_{j \in NB} f_{uj} x_j \geq f_{u0}, \text{ onde } f_{uj} = \bar{a}_{uj} - [\bar{a}_{uj}] \text{ e } f_{u0} = \bar{a}_{u0} - [\bar{a}_{u0}]
\end{aligned} \tag{2.8}$$

Esta desigualdade pode ser reescrita introduzindo uma variável de folga s não-negativa:

$$s = -f_{u0} + \sum_{j \in NB} f_{uj} x_j \tag{2.9}$$

Por exemplo, considere o seguinte problema de programação inteira:

$$\begin{aligned}
z = \max \quad & 4x_1 - x_2 \\
& 7x_1 - 2x_2 \leq 14 \\
& x_2 \leq 3 \\
& 2x_1 - 2x_2 \leq 3 \\
& x_1, x_2 \geq 0, \text{ inteiros}
\end{aligned} \tag{2.10}$$

Adicionando-se as variáveis de folga x_3 , x_4 e x_5 e resolvendo o programa linear, obtem-se:

$$\begin{aligned}
 z = \max \quad & \frac{59}{7} & -\frac{4}{7}x_3 - \frac{1}{7}x_4 \\
 x_1 \quad & + \frac{1}{7}x_3 + \frac{2}{7}x_4 & = \frac{20}{7} \\
 x_2 \quad & + x_4 & = 3 \\
 & -\frac{2}{7}x_3 + \frac{10}{4}x_4 + x_5 & = \frac{23}{7} \\
 & x_1, x_2, x_3, x_4, x_5 \geq 0
 \end{aligned} \tag{2.11}$$

A solução do problema linear é $x = (\frac{20}{7}, 3, 0, 0, \frac{23}{7}) \notin Z_+^5$. Aplicando o corte proposto à primeira restrição (em que $x_1 = \frac{20}{7}$), tem-se: $\frac{1}{7}x_3 + \frac{2}{7}x_4 \geq \frac{6}{7}$, ou, $s = -\frac{6}{7} + \frac{1}{3}x_3 + \frac{2}{7}x_4$. A adição desta restrição ao modelo impedirá que x_1 seja fracionário.

A nova solução linear é

$$\begin{aligned}
 z = \max \quad & \frac{15}{2} & -\frac{1}{5}x_5 - 3s \\
 x_1 \quad & + s & = 2 \\
 x_2 \quad & -\frac{1}{2}x_5 + s & = \frac{1}{2} \\
 x_3 \quad & -x_5 - 5s & = 1 \\
 x_4 \quad & + \frac{1}{2}x_5 + 6s & = \frac{5}{2} \\
 & x_1, x_2, x_3, x_4, x_5, s \geq 0
 \end{aligned} \tag{2.12}$$

A solução ótima $x = (2, \frac{1}{2}, 1, \frac{5}{2}, 0)$ continua fracionária. Escolhendo a segunda restrição, em que $x_2 = \frac{1}{2}$, o corte de Chvátal-Gomory ficaria: $t = -\frac{1}{2} + \frac{1}{2}x_5$. Adicionando-se esta restrição, e resolvendo o novo problema de programação linear, $x = (2, 1, 2, 2, 1, 0, 0)$, ou seja, a solução ótima do problema de programação linear é também solução ótima para o problema original com variáveis inteiras.

2.4.2.2 Desigualdade Válida Básica para Programação Inteira Mista (*The Basic Mixed Integer Inequality*)

A desigualdade válida de Chvátal-Gomory aplica-se a problemas de programação inteira. Em se tratando de problemas de programação inteira mista, a seguinte *proposição* é feita: seja $X^{\geq} = \{(x, y) \in R_+^1 \times Z^1 : x + y \geq b\}$, e $f = b - [b] > 0$; a desigualdade $x \geq f(\langle b \rangle - y)$ ou $\frac{x}{f} + y \geq \langle b \rangle$ é válida para $X^{\geq}$ ¹¹ (a demonstração pode ser encontrada em Wolsey, 1998).

Da mesma forma, propõe-se que, se $X^{\leq} = \{(x, y) \in R_+^1 \times Z^1 : y \leq b + x\}$, e $f = b - [b] > 0$; então a desigualdade $y \leq [b] + \frac{x}{1+f}$ é válida para $X^{\leq}$.

Por exemplo, considere o espaço de solução inteira $X = P \cap (Z^4 \times R^1)$ do problema P , definido por: $P = \{(y, s) \in R_+^4 \times R_+^1 : 13y_1 + 20y_2 + 11y_3 + 6y_4 + s \geq 72\}$. Dividindo cada termo por 11, seria obtida a expressão: $\frac{13}{11}y_1 + \frac{20}{11}y_2 + \frac{11}{11}y_3 + \frac{6}{11}y_4 + \frac{s}{11} \geq \frac{72}{11}$ sendo válida a aproximação $2y_1 + 2y_2 + y_3 + y_4 + \frac{s}{11} \geq \frac{72}{11}$. Usando a primeira proposição apresentada, com $\langle b \rangle = 7$ e $f = \frac{6}{11}$, a seguinte desigualdade válida poderia ser escrita: $\frac{s}{11} \geq \frac{6}{11}(7 - 2y_1 + 2y_2 + y_3 + y_4)$.

2.4.2.3 Desigualdade Válida para Arredondamento em Programação Inteira Mista (*The Mixed Integer Rounding Inequality*)

Seja X^{MIR} um conjunto viável com a seguinte estrutura: $X^{MIR} = \{(x, y) \in R_+^1 \times Z_+^2 : a_1y_1 + a_2y_2 \leq b + x\}$, onde a_1, a_2 e b são escalares e $b \notin Z^1$.

¹¹ A generalização para $Y \in Z_+^p$, $p > 1$ é válida. O autor propõe um exemplo com $p=4$.

Wolsey (1998) demonstra a veracidade da proposição: Seja $f = b - [b]$ e $f_i = a_i - [a_i]$ para $i=1, 2$; suponha também que $f_1 \leq f \leq f_2$; então $[a_1]y_1 + \left([a_2] + \frac{f_2 - f}{1 - f}\right)y_2 \leq [b] + \frac{x}{1 - f}$ é válida para X^{MIR} ¹².

Por exemplo, considere o conjunto $X = \{(y, x) \in Z_+^2 \times R_+^1 : \frac{10}{3}y_1 + \frac{11}{4}y_2 \leq \frac{21}{2} + x\}$. Sendo $f = 1/2, f_1 = 1/3, f_2 = 3/4$ a desigualdade $3y_1 + \frac{5}{2}y_2 \leq 10 + 2x$ é válida para X .

2.4.2.4 Desigualdades Disjuntivas (Disjunctive Inequalities)

O conjunto $X = X^1 \cup X^2$ com $X^i \subseteq R_+^n$, para $i=1, 2$, é uma disjunção de dois conjuntos X^1 e X^2 . Wolsey (1998) demonstra que a seguinte proposição é verdadeira: se $\sum_{j=1}^n \pi_j^i x_j \leq \pi_0^i$ for válida para X^i , $i=1, 2$, então a desigualdade

$\sum_{j=1}^n \pi_j x_j \leq \pi_0$ é válida para X , se $\pi_j \leq \min[\pi_j^1, \pi_j^2]$ para $j=1, \dots, n$ e $\pi_0 \geq \max[\pi_0^1, \pi_0^2]$;

2.4.2.5 Desigualdades de Cobertura (Cover Cuts)

Alguns problemas poderão conter restrições com estruturas semelhantes ao problema da mochila binária. Considere o conjunto $X = \left\{x \in B^n : \sum_{j=1}^n a_j x_j \leq b\right\}$, e suponha que os coeficientes $\{a_j\}_{j=1}^n$ sejam positivos e que $b > 0$; seja $N = \{1, \dots, n\}$. O conjunto $C \subseteq N$ é uma cobertura do conjunto X se $\sum_{j \in C} a_j \geq b$. A cobertura será mínima, se $C \setminus \{j\}$ não for uma cobertura para qualquer $j \in C$. Se $C \subseteq N$ for uma

¹² A generalização para $Y \in Z_+^p$, $p > 1$ é válida. O autor propõe um exemplo com $p=3$.

cobertura, então a desigualdade de cobertura $\sum_{j \in C} x_j \leq |C| - 1$ é válida para X (ver a demonstração em Wolsey; 1998).

Por exemplo, considerando o conjunto do problema da mochila binária $X = \{x \in B^7 : 11x_1 + 6x_2 + 6x_3 + 5x_4 + 5x_5 + 4x_6 + x_7 \leq 19\}$. Algumas restrições de cobertura, que configuram-se como desigualdades válidas são:

$$\begin{aligned} x_1 + x_2 + x_3 &\leq 2 \\ x_1 + x_2 + x_6 &\leq 2 \\ x_1 + x_5 + x_6 &\leq 2 \\ x_3 + x_4 + x_5 + x_6 &\leq 3 \end{aligned} \tag{2.13}$$

2.4.2.6 Cortes de Limitantes Implícitos (Implied Bounds Cuts)

Um tipo de restrição que pode aparecer em alguns problemas tem a seguinte estrutura: $x_1 + x_2 + \dots + x_n \leq (u_1 + u_2 + \dots + u_n)y$ $y \in \{0,1\}$, onde u_j é um limitante superior válido para $x_j \geq 0$, $j = 1, \dots, n$. Esta restrição é equivalente, em função da integralidade de y , à seguinte coleção de restrições desagregadas: $x_j \leq u_j y$ ($j = 1, \dots, n$).

2.4.2.7 Cortes por Incompatibilidade (Clique Cuts)

Cortes por incompatibilidade são restrições que impedem que diversas variáveis (binárias) de decisão assumam valor igual a 1 simultaneamente. Assim, dentre uma coleção de variáveis, no máximo uma pode ser escolhida e esta escolha é mutuamente exclusiva. Diz-se, portanto, que as variáveis são incompatíveis entre si.

As variáveis binárias x e y são consideradas incompatíveis se: $x + y \leq 1$. Um *clique* é um conjunto de variáveis $C = \{x \in B^n : x_i \text{ e } x_j \text{ são incompatíveis, } \forall i, j\}$. Um

corte de incompatibilidade é dado por: $\sum_{j \in C} x_j \leq 1$. Por exemplo, se $x + y \leq 1; x + z \leq 1; y + z \leq 1$, então $x + y + z \leq 1$.

2.4.2.8 Desigualdades de Cobertura de Fluxo (Flow Cover Inequalities)

Alguns problemas de programação inteira mista podem ter a estrutura:

$$X = \left\{ (x, y) \in R_+^n \times B^n : \sum_{j \in N_1} x_j \leq b + \sum_{j \in N_2} x_j, \quad x_j \leq a_j y_j \text{ para } j \in N_1 \cup N_2 \right\}, \text{ que assemelha-se}$$

às estruturas dos problemas de fluxo em rede com custos fixos.

O conjunto $C = C_1 \cup C_2$ com $C_1 \subseteq N_1, C_2 \subseteq N_2$ é uma cobertura generalizada para X se $\sum_{j \in C_1} a_j - \sum_{j \in C_2} a_j = b + \lambda$, com $\lambda > 0$; λ é chamado de excesso da cobertura.

Wolsey (1998) demonstra que a seguinte proposição (desigualdade de cobertura de fluxo) é verdadeira para X , em que $L_2 \subseteq N_2 \setminus C_2$:

$$\sum_{j \in C_1} x_j + \sum_{j \in C_1} (1 - y_j) (a_j - \lambda)^+ \leq b + \sum_{j \in C_2} a_j + \lambda \sum_{j \in L_2} y_j + \sum_{j \in N_2 \setminus (C_2 \cup L_2)} x_j \quad (2.14)$$

Por exemplo, seja o conjunto:

$$X = \left\{ (x, y) \in R_+^6 \times B^6 : x_1 + x_2 + x_3 \leq 4 + x_4 + x_5 + x_6, \right. \\ \left. x_1 \leq 3y_1, x_2 \leq 3y_2, x_3 \leq 6y_3, x_4 \leq 3y_4, x_5 \leq 5y_5, x_6 \leq 1y_6 \right\} \quad (2.15)$$

Fazendo $C_1 = \{1, 3\}$ e $C_2 = \{4\}$, $C = (C_1, C_2)$ será uma cobertura generalizada com $\lambda = 2$. Fixando $L_2 = \{5\}$, a desigualdade válida de cobertura de fluxo resultante será: $x_1 + x_3 + 1(1 - y_1) + 4(1 - y_3) \leq 7 + 2y_5 + x_6$.

2.4.2.9 *Desigualdade por Cobertura de Caminhos de Fluxo (Flow Path Cuts)*

Roy; Wolsey (1985) estudaram o problema de fluxo em rede com custos fixos, em que os arcos da rede estão sujeitos a decisões de ‘abrir’ e ‘fechar’. Este problema permite incorporar os aspectos presentes na estrutura de problemas de definição de lotes de produção e em problemas de distribuição. Os autores propõem uma desigualdade válida para tratar a seguinte situação: caso o fluxo em um conjunto de arcos seja nulo, então o fluxo nos arcos que alimentam estes arcos de fluxo nulo também deverá ser igual a zero.

2.4.2.10 *Desigualdade de Cobertura GUB (GUB Cuts)*

Uma outra categoria de desigualdade válida que tem se mostrado eficiente (Williams, 1999) é resultante da identificação de limitantes que são comuns a um conjunto de variáveis em problemas de programação inteira, ou inteira mista. Enquanto algumas variáveis têm limitantes do tipo $x \leq M$, a desigualdade da classe GUB (*generalized upper bound*) procura identificar um conjunto de variáveis que, somadas, tenham limite superior igual a b . Assim, esta restrição teria a seguinte estrutura: $\sum_j x_j \leq b$ ou $\sum_j x_j = b$.

2.4.3 *Branch and Cut*

Quando o número de desigualdades válidas (VI) necessárias para restringir adequadamente o espaço de solução é muito grande, a adição de todas estas desigualdades pode dificultar a resolução do problema relaxado. Baseado no fato que para resolver o problema relaxado à otimalidade em P_x são necessárias no máximo $n + p$ desigualdades, pode-se fazer uso das desigualdades de maneira

alternativa¹³. Ao invés de adicionar inicialmente todas as desigualdades válidas (VI) de C à formulação do problema, utiliza-se a formulação inicial P_X e o problema relaxado LR é resolvido. Em seguida, à medida que algumas desigualdades válidas forem violadas na solução ótima de LR , estas são incorporadas à formulação.

O processo de identificação de quais desigualdades da família C foram violadas para uma solução (x^*, y^*) é chamado de “problema de separação” $SEP((x^*, y^*)|C)$, que consiste em verificar se $(x^*, y^*) \in C$. Esta conferência é feita por um algoritmo de separação, que avalia cada desigualdade de C . Estes algoritmos podem ser exatos, quando garantem que toda desigualdade violada será identificada, ou heurísticos, quando a conferência é simplificada visando a redução do tempo de processamento.

O algoritmo de planos de corte, mostrado na figura 2.2, apresenta o mecanismo de separação, resolvendo o problema relaxado P_X sem adicionar, *a priori*, as desigualdades válidas (VI) de C . A solução relaxada obtida é submetida ao processo de separação, que identifica as desigualdades violadas, incorporando-as à formulação corrente V , $V \subseteq P_X$.

O método “Branch and Cut” (BC) é uma extensão do método B&B, o qual inclui uma família C de desigualdades válidas. Ao selecionar um determinado problema V da lista L para ser resolvido (passo 3 do algoritmo B&B), a solução obtida pela resolução do problema de programação linear é submetida ao processo de separação, visando introduzir restrições que limitem o espaço de solução,

¹³ $X = \{(x, y) \in R_+^n \times Z_+^p : Ax + By \geq b\}$ - é o conjunto de soluções viáveis de um MIP;

$P_X = \{(x, y) \in R_+^n \times R_+^p : Ax + By \geq b\}$ - é o problema de programação linear relaxado associado;

n - número de variáveis reais de um MIP;

p - número de variáveis inteiras de um MIP;

possibilitando que limitantes inferiores de melhor qualidade sejam obtidos, ao mesmo tempo que o espaço de solução é reduzido. O BC, listado na figura 2.3, é uma alternativa à estratégia de reformulação *a priori*, quando todas as VIs são adicionadas previamente.

Algoritmo de Planos de Corte

```

a. Inicializar  $W:=V$ 
   Calcular  $Z(W)$ , a solução ótima do problema de programação linear relaxado,
   sem incorporar VIs de  $C$ 
   Seja  $(x^*, y^*)$  a solução ótima relaxada
b. Resolver o problema de separação  $SEP((x^*, y^*) | C)$ 
   se todas as desigualdades em  $C$  são satisfeitas em  $(x^*, y^*)$ , então
        $(x^*, y^*) \in V \cap C$  e é solução ótima com valor  $Z(V \cap C)$ 
       fim
   se não // SEP devolveu alguma desigualdade violada  $(\alpha^j x + \beta^j y \geq \gamma^j)$ 
       Adicionar  $(\alpha^j x + \beta^j y \geq \gamma^j)$  à formulação:
        $W := W \cap \{(x, y) \in R^n \times R^p \mid \alpha^j x + \beta^j y \geq \gamma^j\}$ 
       Calcular a solução ótima relaxada  $Z(W)$ 
       Seja  $(x^*, y^*)$  a solução ótima relaxada
       Ir para b
fim_se

```

Figura 2.2 – Pseudo código do algoritmo de Planos de Corte (adaptado de Pochet; Wolsey, 2006)

2.4.4 Heurísticas de Construção

As heurísticas construtivas possuem um papel importante no processo de resolução de problemas de programação inteira mista (MIP), ao tentarem gerar soluções viáveis – eventualmente, de alta qualidade – em um curto espaço de tempo. Caso uma solução seja obtida, esta poderá ser objeto de melhoria por meio de um método apropriado. Três métodos construtivos passam a ser descritos.

Algoritmo Branch and Cut

1. Inicialização

$$L = \{P_X\}$$

$$\bar{Z} = +\infty$$

Assumir que LR é limitado ($Z(P_X) > -\infty$)

2. Término

se $L = \emptyset$, **então**:

se $\bar{Z} = +\infty$, **então** $X = \emptyset$ (problema inviável)

se $\bar{Z} < +\infty$, **então** a solução $(x, y) \in X$ com $\bar{Z} = cx + fy$ é ótima

fim

3. Seleção de um nó e Solução do problema com Algoritmo de Plano de Corte

Selecionar $V \in L$ e fazer $L = L \setminus \{V\}$

3a. Achar a solução ótima (x^V, y^V) e o valor $Z(V)$ do problema V

3b. Resolver o problema de separação $SEP((x^V, y^V) | C)$

se todas as desigualdades em C são satisfeitas em (x^V, y^V) , **então**

$(x^V, y^V) \in V \cap C$ é solução ótima com valor $Z(V \cap C)$

ir para o passo 4

se não // SEP devolveu alguma desigualdade violada

adicionar $(\alpha^j x + \beta^j y \geq \gamma^j)$ à formulação:

$$V := V \cap \{(x, y) \in R^n \times R^p \mid \alpha^j x + \beta^j y \geq \gamma^j\}$$

Calcular a solução ótima relaxada $Z(V)$ e a solução (x^V, y^V)

Seja (x^*, y^*) a solução ótima relaxada

ir para o passo 3b

4. Eliminação

se $Z(V) \geq \bar{Z}$, **então** ir para o passo 2 (V é inviável ou dominada pela melhor solução inteira já encontrada)

se $Z(V) < \bar{Z}$, **então**

se $y^V \in Z^p$, **então** (a solução é integral)

atualizar o limitante superior fazendo $\bar{Z} = Z(V)$

atualizar a lista L removendo os problemas dominados

(para cada $W \in L$: **se** $Z(W) \geq \bar{Z}$, **então** $L = L \setminus \{W\}$)

ir para o passo 2

5. Ramificação

(ocorre apenas quando V não foi eliminado: $Z(V) < \bar{Z}$ e $y^V \notin Z^p$)

Selecionar j tal que $y_j^V \notin Z$

Atualizar a lista L :

$$L = L \cup \{V^0, V^1\} \text{ onde:}$$

$$V^0 = V \cap \{(x, y) \in R_+^n \times R_+^p : y_j \leq \lfloor y_j^V \rfloor\}$$

$$V^1 = V \cap \{(x, y) \in R_+^n \times R_+^p : y_j \geq \lceil y_j^V \rceil\}$$

Ir para o passo 2

Figura 2.3 – Pseudo código do algoritmo “Branch and Cut” (adaptado de Pochet;

Wolsey, 2006)

2.4.4.1 LP-and-Fix¹⁴

Esta heurística realiza a busca em profundidade na árvore BC. A estratégia geral deste método consiste, em cada nó da árvore “branch and bound”, fixar as variáveis y que são inteiras na solução do problema de programação linear relaxado, e escolher uma das variáveis não inteiras para proceder com o processo de ramificação.

A orientação para busca em profundidade é chamada de mergulho (“dives”). A versão conhecida como “LP-driven dives” usa a última solução linear $(\hat{x}, \hat{y})$ e fixa a variável menos fracionária. Seja $F = \{j : \hat{y}_j \notin Z^1\}$. Este algoritmo determina o elemento $g_k = \min_{j \in F} g_j$, onde $g_j = \min\{\hat{y}_j, 1 - \hat{y}_j\}$ e impõe $y_k=0$ se $\hat{y}_k \leq 0,5$ e $y_k=1$, caso contrário.

O algoritmo “LP-and-Fix” é uma variante do “LP-driven dives”. O problema de programação linear relaxado é resolvido e todas as variáveis inteiras (do problema original) cujos valores são inteiros são fixadas. Em seguida, o problema resultante (MIP^{LP-FIX}) é resolvido novamente:

$$\min\{cx + fy : Ax + By \geq b, x \in R_{n+}, y \in \{0,1\}^p, y_j = \hat{y}_j \text{ para todo } j \in Q \text{ com } \hat{y}_j \in \{0,1\}\}$$

Caso a solução deste novo problema indique solução inviável, a heurística falhou e deverá ser interrompida. Caso contrário, uma solução heurística terá sido encontrada.

2.4.4.2 Relax-and-Fix

Este algoritmo pressupõe que as variáveis binárias do problema podem ser particionadas em R conjuntos disjuntos $Q^1, \dots, Q^R$ de importância decrescente. São

¹⁴ Para os métodos *LP-and-Fix* e *Relax-and-Fix*, o conjunto $Q = \{1, \dots, p\}$ designará o conjunto dos índices das variáveis binárias y .

também criados R subconjuntos U^r envolvendo as variáveis binárias, de forma que $U^r \subseteq \bigcup_{u=r+1}^R Q^u$, para $r = 1, \dots, R-1$.

Em primeiro lugar, o problema MIP^1 é resolvido impondo a restrição de integralidade apenas para as variáveis que pertençam a $Q^1 \cup U^1$. O MIP^1 apresenta a seguinte estrutura:

$$\begin{aligned} \min \{ & cx + fy : Ax + By \geq b \\ & x \in R_+^n \\ & y_j \in \{0,1\} \quad \forall j \in Q^1 \cup U^1 \\ & y_j \in [0,1] \quad \forall j \in Q \setminus (Q^1 \cup U^1) \} \end{aligned} \quad (2.16)$$

Seja (x^1, y^1) a solução ótima do problema MIP^1 . As variáveis em Q^1 são fixadas em y^1 , e procede-se com a resolução do problema MIP^2 . Nas iterações seguintes MIP^r , $2 \leq r \leq R$, os valores de y em Q^{r-1} são fixados em seus valores ótimos, obtidos em MIP^{r-1} e impõe-se a restrição de integralidade para as variáveis importantes pertencentes à união $Q^r \cup U^r$. O problema genérico MIP^r tem a estrutura indicada a seguir.

$$\begin{aligned} \min \{ & cx + fy : \\ & Ax + By \geq b \\ & x \in R_+^n \\ & y_j = y_j^{r-1} \in \{0,1\} \quad \forall j \in Q^1 \cup \dots \cup Q^{r-1} \\ & y_j \in \{0,1\} \quad \forall j \in Q^r \cup U^r \\ & y_j \in [0,1] \quad \forall j \in Q \setminus (Q^1 \cup \dots \cup Q^r \cup U^r) \} \end{aligned} \quad (2.17)$$

Seja (x^r, y^r) seja a solução ótima de MIP^r $2 \leq r \leq R$. Ou MIP^r será inviável para algum $r \in \{1, \dots, R\}$ e a heurística falhou, ou (x^R, y^R) será a solução heurística do problema MIP original.

2.4.4.3 Feasibility Pump

Este método foi desenvolvido por Fischetti, et al. (2005) e posteriormente aperfeiçoado por Achterberg; Berthold (2005). A notação, em seguida apresentada, será usada para introduzir a idéia central destes autores. Um problema MIP pode ser representado por:

$$\begin{aligned}
 (MIP) \quad & \min c^T x \\
 & Ax \geq b \\
 & x_j \geq 0, \text{ inteiro} \quad \forall j \in I \\
 & x_j \geq 0 \quad \quad \quad \forall j \in C
 \end{aligned} \tag{2.18}$$

Seja $P = \{x : Ax \geq b, x \geq 0\}$ o poliedro associado ao problema de programação linear relaxado do MIP. O vetor x será dito inteiro se x_j for inteiro, $\forall j \in I$. O vetor $\tilde{x}$ será o arredondamento de x , realizado da seguinte forma: $\tilde{x}_j = [x_j]$, se $j \in I$; em caso contrário, $\tilde{x}_j = x_j$, sendo $[.]$ o operador de arredondamento para o inteiro mais próximo. Para indicar a distância de um ponto $x \in P$ a um ponto $\tilde{x}$ será utilizada a função:

$$\Delta(x, \tilde{x}) = \sum_{j \in I} |x_j - \tilde{x}_j| \tag{2.19}$$

Considerando que as variáveis $x_j \in I$ possuem como limitantes inferior e superior, respectivamente, os parâmetros l_j e u_j , isto é, $l_j \leq x_j \leq u_j$, a função distância pode ser escrita conforme apresentada:

$$\Delta(x, \tilde{x}) = \sum_{j \in I: \tilde{x}_j = l_j} (x_j - l_j) + \sum_{j \in I: \tilde{x}_j = u_j} (u_j - x_j) + \sum_{j \in I: l_j < \tilde{x}_j < u_j} (x_j^+ + x_j^-) \tag{2.20}$$

Nesta formulação (2.20), novas variáveis foram introduzidas, a saber, x_j^+ e x_j^- , as quais requerem que novas restrições sejam incorporadas ao modelo original:

$$x_j = \tilde{x}_j + x_j^+ - x_j^- \quad x_j^+ \geq 0, \quad x_j^- \geq 0 \quad \forall j \in I : l_j < \tilde{x}_j < u_j \quad (2.21)$$

Assim, dado um ponto inteiro $\tilde{x}$, o ponto mais próximo $x^* \in P$ pode ser determinado resolvendo-se o problema de programação linear dado por:

$$\min\{\Delta(x, \tilde{x}) : Ax \geq b\} \quad (2.22)$$

Se a distância $\Delta(x^*, \tilde{x}) = 0$, então $x_j^*(= \tilde{x}_j)$ será inteiro para todo $j \in I$ e x^* será uma solução viável para o problema MIP. Além disso, dado um ponto $x^* \in P$, o ponto inteiro $\tilde{x}$ mais próximo de x^* poderá ser obtido pelo processo de arredondamento descrito.

O método “feasibility pump” inicia em um ponto qualquer $x^* \in P$ (usualmente, x^* é a solução do problema de programação linear relaxado), tendo como solução arredondada $\tilde{x}$, em geral, um ponto inviável para o MIP original. Se $\Delta(x^*, \tilde{x}) = 0$, então x^* será uma solução viável para o MIP, e o método encerra. Caso contrário, $\tilde{x}$ é substituído pelo arredondamento de x^* , e o seguinte problema de programação linear é resolvido: $x^* = \min\{\Delta(x, \tilde{x}) : Ax \geq b\}$. Se a distância obtida $\Delta(x^*, \tilde{x}) > 0$ então o processo de arredondamento volta a ocorrer. O principal problema na implementação deste método é a possibilidade de ocorrência de ciclagem. Os autores sugerem, como forma de contornar este problema, a introdução de uma perturbação na solução corrente, fazendo com que o processo de busca (e convergência) seja reiniciado a partir de uma nova solução x^* .

A melhoria sugerida por Achterberg; Berthold (2005) consiste em, ao invés de descartar a função objetivo original do problema, conforme consta em (2.22), fazer com a sua influência seja progressivamente diminuída, dando lugar à função objetivo

de distância. Desta maneira, aumenta-se a possibilidade de convergência para soluções viáveis que sejam de alta qualidade.

A função objetivo (2.19) passaria a ter a seguinte estrutura:

$$\Delta_{\alpha}(x, \tilde{x}) := (1 - \alpha)\Delta(x, \tilde{x}) + \alpha \frac{\|\Delta(x, \tilde{x})\|}{\|c\|} c^T x \quad \alpha \in [0, 1] \quad (2.23)$$

Nesta expressão, $\|\cdot\|$ é a norma euclidiana do vetor; e o peso α é geometricamente reduzido, a cada iteração, segundo um parâmetro fixo $\varphi < 1$, ou seja, $\alpha_{t+1} = \varphi\alpha_t$ e $\alpha_0 \in [0, 1]$.

2.4.5 Heurísticas de Melhoria

As heurísticas, em seguida descritas, buscam a solução ótima por meio de um processo contínuo de melhoria. Todas elas se baseiam na estrutura matemática do problema de programação inteira mista (MIP) e direcionam a forma como o método de enumeração implícita “branch and bound” explora o espaço de solução. Em função disto, as mesmas interagirão com pacotes de otimização matemática, tais como o CPLEX e o XPRESS.

2.4.5.1 Relaxation Induced Neighborhood Search – “RINS”

Proposto por Danna et al. (2005), a estratégia de melhoria “RINS” explora o subespaço de solução que une a solução do problema de programação linear relaxado $(\hat{x}, \hat{y})$ a uma solução inteira viável conhecida $(\bar{x}, \bar{y})$. As variáveis y_j que tiverem os mesmos valores em ambas soluções são fixadas. Em seguida, é resolvido o problema MIP^{RINS} definido como:

$$\begin{aligned}
\min \{ & cx + fy : Ax + By \geq b \\
& x \in R_+^n, y \in \{0,1\}^p \\
& y_j = \bar{y}_j \in \{0,1\} \quad \forall j \in Q \text{ com } \bar{y}_j = \hat{y}_j \}
\end{aligned} \tag{2.24}$$

A solução obtida, caso seja viável quanto à integralidade, terá definido um novo limitante para a solução do problema. Este procedimento é iterado até que a solução ótima seja obtida.

2.4.5.2 Local Branching

A técnica “local branching” (LB) foi desenvolvida por Fischetti; Lodi (2003), como uma alternativa para resolver problemas MIP de grande porte. A técnica LB gerencia ou delimita, em um nível ‘estratégico’, uma sub-região do espaço de solução que será explorada, no nível ‘tático’, por um software de otimização matemática genérico. A busca em regiões menores possibilita que novas soluções sejam encontradas de maneira mais fácil e rápida, quando comparado com o desempenho dos softwares de otimização existentes, que exploram todo o espaço de solução.

A estrutura genérica a ser empregada para o problema MIP é:

$$\begin{aligned}
(P) \quad & \min c^T x \\
& Ax \geq b \\
& x_j \in \{0,1\} \quad \forall j \in B \neq \emptyset \\
& x_j \geq 0, \text{ inteiro} \quad \forall j \in G \\
& x_j \geq 0 \quad \forall j \in C
\end{aligned} \tag{2.25}$$

O conjunto de índice das variáveis $N=\{1, 2, \dots, n\}$ é particionado em três subconjuntos (B, G, C) correspondendo, respectivamente, às variáveis binárias, inteiras e reais.

Uma sub-região é construída na vizinhança de uma solução existente, da seguinte forma: dada uma solução de referência $\bar{x}$ de (P) , seja $\bar{S} = \{j \in B : \bar{x}_j = 1\}$, designado como ‘suporte binário’ de $\bar{x}$. Para um parâmetro k informado, define-se como vizinhança k -OPT de $\bar{x}$, ou $N(\bar{x}, k)$, o conjunto de soluções viáveis de (P) que satisfazem à restrição de ramificação local (“local branching constraint”):

$$\Delta(x, \bar{x}) := \sum_{j \in \bar{S}} (1 - x_j) + \sum_{j \in B \setminus \bar{S}} x_j \leq k \quad (2.26)$$

Esta restrição faz com que, no máximo, k variáveis binárias sejam exploradas simultaneamente, definindo uma vizinhança de tamanho 2^k . O parâmetro k deverá ser suficientemente pequeno de forma a garantir que a vizinhança seja explorada rapidamente. Ao mesmo tempo, deverá ser grande o suficiente para que a vizinhança contenha soluções melhores que $\bar{x}$.

O algoritmo LB é precedido pelo algoritmo “branch and bound”, com o intuito de que este gere uma solução inicial $\bar{x}^1$ para o problema MIP¹⁵. Em seguida, uma nova restrição é adicionada para reduzir o espaço de solução X , usando a função distância $\Delta(x, \bar{x}^1)$, formando o conjunto $X_1 = X \cap N(\bar{x}^1, k)$. Se uma solução $\bar{x}^2$ for encontrada com o valor da função objetivo melhorado, dentro de um limite de tempo de processamento *lim_nó*, então um novo problema é gerado, em que a desigualdade $\leq$ da restrição (2.26) é substituída por $\geq k+1$, e uma nova restrição do tipo (2.26) é acrescentada, tendo como base a solução $\bar{x}^2$ ($\Delta(x, \bar{x}_2) \leq k$). Assim, o novo espaço de solução X_2 para este problema fica definido como:

$$X_2 = X \cap (N(\bar{x}^2, k) \setminus N(\bar{x}^1, k)) = (X \setminus X_1) \cap N(\bar{x}^2, k) \quad (2.27)$$

¹⁵ Muitas vezes, é conveniente informar a solução inicial, gerando-a através de um procedimento heurístico.

O algoritmo “branch and bound” volta a ser executado com o intuito de encontrar novas soluções. Este procedimento de ramificação será iterado enquanto houver melhoria nos valores da função objetivo, respeitando o limite de tempo $lim_nó$. Após l iterações, l restrições terão sido acrescentadas e o espaço de solução terá a seguinte estrutura:

$$X_l = (X \setminus X_1 \setminus X_2 \dots \setminus X_{l-1}) \cap N(\bar{x}^l, k) \quad (2.28)$$

Se a solução não for melhorada no tempo $lim_nó$, a vizinhança de X_l é reduzida, substituindo k da restrição (2.26) por $k/2$ ($\Delta(x, \bar{x}^l) \leq k/2$), no passo chamado de intensificação. Caso haja a constatação de inviabilidade ou, se durante o limite de tempo $lim_nó$ uma solução viável não for encontrada, ocorre o passo de diversificação, em que o lado direito da restrição (2.26) é aumentado em $k/2$ ($\Delta(x, \bar{x}^l) \leq k + k/2$), uma restrição do tipo $\Delta(x, \bar{x}^l) > 1$ é introduzida (visando forçar a exploração desta vizinhança) e todas as demais restrições de ramificação são removidas. O parâmetro dv_max limita o número de diversificações que poderão ser executadas durante a busca. Por último, controlando o algoritmo tem-se o parâmetro $tmax$, limitando a duração da rotina LB.

2.4.5.3 Variable Neighborhood Search

A técnica de busca em vizinhança variável (VNS – “variable neighborhood search”) foi proposta por Mladenović; Hansen (1997), sendo posteriormente descrita nos trabalhos de Hansen; Mladenović (2001, 2003) e Hansen et al. (2006).

A inovação apresentada pelos autores está em combinar a busca local com mudanças sistemáticas da estrutura da vizinhança, de forma a escapar de regiões de ótimo local. Esta redefinição visa à exploração de vizinhanças cada vez mais distantes de uma solução de referência. A busca em vizinhança variável se dá pela

geração aleatória de soluções nas diversas vizinhanças de uma solução, seguidas de busca local. Assim que uma solução melhorada é encontrada, esta passa a ser a nova solução de referência. A figura 2.4 ilustra conceitualmente os passos do VNS.

Inicialização. Selecionar um conjunto de estruturas de vizinhança N_k $k=1, \dots, k_{max}$, que serão usadas na busca; 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''
- (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 2.4 – Estrutura geral da busca em vizinhança variável (adaptado de Hansen; Mladenović, 2001)

Diversas variações da estrutura geral do VNS são possíveis. Uma delas é a busca descendente em vizinhança variável (VND – “variable neighborhood descent”), em que, ao invés de serem geradas diversas soluções aleatórias nas vizinhanças de uma solução corrente, para posterior melhoria, realiza-se a busca local nas diversas estruturas de vizinhança tendo como referência uma única solução, a solução corrente. Este método está apresentado na figura 2.5.

Inicialização. Selecionar um conjunto de estruturas de vizinhança N_k $k=1, \dots, k_{max}$, que serão usadas na busca; informar uma solução inicial x ;

Até que não haja mais melhorias, *Repetir*:

(1) Fazer $k \leftarrow 1$; (2) Até que $k = k_{max}$, repetir os passos:

- (a) *Busca Local.* Encontrar melhor vizinho x' de x ($x' \in N_k(x)$)
- (b) *Mover ou não.* Se a solução obtida x' for melhor que x , fazer $x \leftarrow x'$; caso contrário, fazer $k \leftarrow k+1$;

Figura 2.5 – Estrutura geral da busca descendente em vizinhança variável (adaptado de Hansen; Mladenović, 2001)

A técnica VNS vem sendo empregada de maneira crescente, embora ainda modesta quando comparada com as metaheurísticas convencionais (busca tabu, algoritmo genético, “simulated annealing”, entre outras). Alguns relatos de sucesso na aplicação são dados por: Hansen; Mladenović (2001), resolvendo problemas clássicos como o do caixeiro viajante, o problema das p -medianas, o problema de agrupamento ou clusterização (“minimum sum-of-squares clustering problem”), entre outros. Prandtstetter; Raidl (2005) conseguiram melhorar o “benchmark” para problemas de sequenciamento em linha de produção (sistema contínuo) de veículos, sujeito a set-ups. Um recente congresso europeu de pesquisa operacional foi organizado para discutir a aplicabilidade do VNS a diversos problemas reais.¹⁶

A estrutura apresentada para o VNS traz consigo a possibilidade de utilização desta técnica acoplada a softwares de otimização matemática, na etapa de geração de solução inicial e, principalmente, no processo de busca local. Usando como base a formulação matemática do problema e a definição de vizinhança apresentada na técnica de “local branching”, é possível gerar diversas vizinhanças tão-somente variando o parâmetro k , indicativo do tamanho e, portanto, do distanciamento, do espaço de solução em relação à solução corrente.

2.5 Conclusão da Revisão Bibliográfica

A revisão bibliográfica permitiu ter uma visão geral dos problemas de planejamento em transporte, com ênfase em transporte marítimo. O problema abordado nesta pesquisa, embora sendo uma operação de apoio “offshore” e não específica de transporte de carga, enquadra-se na modalidade de operação industrial. Trata-se de uma situação em que uma empresa dispõe de um conjunto de

¹⁶ Consultar o sítio <http://www.mecvns.com>

embarcações e necessita programar esta frota, sobre a qual detém o controle operacional, designando-a às diversas tarefas.

Desrochers et al. (1990) apresentaram uma taxonomia que evidenciou a principal restrição operacional do problema desta pesquisa, que é a requisição simultânea de veículos para execução de uma tarefa. Esta característica difere das aplicações conhecidas de problemas de programação de veículos, devendo limitar a aplicabilidade dos métodos existentes para solucionar o problema desta pesquisa, e requer uma abordagem específica.

Quanto à representação matemática utilizada, a maior parte dos autores optou por incorporar as características particulares de seus problemas de maneira implícita, isto é, através de rotas viáveis geradas por meio de procedimentos heurísticos¹⁷. Assim, grande parte dos modelos visam selecionar um subconjunto de rotas que garantem o atendimento da demanda – em geral, problemas de partição de conjuntos – deixando os aspectos operacionais como janela de tempo, precedência, compatibilidade veículo x cliente, capacidade de carga, entre outros, a cargo das rotinas de construção de rotas. Estas abordagens permitem a obtenção de soluções ótimas dentre o conjunto de rotas propostas, sendo aplicáveis quando o cenário é composto por um número pequeno de tarefas. A presente pesquisa refere-se a um problema com até 60 tarefas, o que torna inviável a enumeração de todas as rotas possíveis. Além disso, o processo de construção de rotas torna-se bastante complexo uma vez que a demanda por múltiplas embarcações (de uma frota heterogênea, com possibilidade de cessão de embarcações entre classe) obriga que a programação das embarcações e das tarefas respeite o aspecto temporal, isto é, a simultaneidade no atendimento das requisições.

¹⁷ Cabe ressaltar que alguns procedimentos para geração de rotas preservam a otimalidade, como no caso no método de geração de colunas.

As exceções observadas na revisão bibliográfica quanto à enumeração completa das rotas foram Brønmo et al. (2005) e Sherali et al. (1999). Este último trabalho se destaca por apresentar uma formulação matemática complexa, baseada em fluxo em rede, que servirá de referência para a presente pesquisa. Estes autores também aplicaram desigualdades válidas para acelerar o processo de exploração “branch and bound”. O conceito introduzido por Fagerholt (2000), de incorporação do nível de serviço na função objetivo do problema, alinha-se a um aspecto importante desta pesquisa, em que a priorização no atendimento das requisições críticas (isto é, com elevadas penalidades) deve ser observada.

Quanto aos algoritmos revisados, observou-se que o emprego de heurísticas construtivas é interessante por auxiliar na obtenção de soluções iniciais para o problema, que poderá ser, posteriormente, objeto de melhoria.

Após a etapa de revisão da literatura, pretende-se, como próximos passos:

1. Elaborar representações matemáticas do problema desta pesquisa;
2. Desenvolver heurísticas construtivas capazes de gerar soluções iniciais viáveis baseadas na estrutura matemática do problema (por exemplo, o método “feasibility pump”), e heurísticas derivadas a partir de regras de decisão eficazes, que explorem as características do problema;
3. Utilizar o método “branch and cut” que acompanha o pacote de otimização CPLEX 10.0 para obter soluções ótimas para os problemas de menor porte, e gerar soluções viáveis para os problemas de maior porte, que venham ser submetidas a procedimentos de melhoria;
4. Utilizar as heurísticas de melhoria “local branching” e “variable neighborhood search” para buscar soluções melhores na vizinhança das soluções geradas pelos procedimentos construtivos.

3 MODELAGEM MATEMÁTICA

O presente capítulo apresenta as formulações matemáticas representativas do problema de PROGRAMAÇÃO DE FROTA DE APOIO A OPERAÇÕES “OFFSHORE” SUJEITA À REQUISIÇÃO DE MÚLTIPLAS EMBARCAÇÕES PARA UMA MESMA TAREFA. Após uma descrição detalhada do problema, os modelos matemáticos serão apresentados e discutidos.

3.1 Descrição Detalhada do Problema

O problema desta pesquisa retrata uma operação de serviço de reboque de plataformas marítimas e lançamento de linhas de ancoragem realizada por embarcações especiais, em um horizonte de curto a médio prazo (3 a 6 meses). As características inerentes a esta operação remetem aos problemas de programação de veículos, e consistem em: minimizar o custo variável de uma frota e a parcela associada ao nível de serviço, ao atender integralmente a demanda, que é composta por tarefas que deverão iniciar dentro de janelas de tempo previstas, respeitando restrições de precedência. Há imposições de compatibilidade entre as embarcações e as tarefas, havendo diversas embarcações aptas para atender à solicitação de serviço. Múltiplas embarcações são requisitadas por tarefa, sendo o início das tarefas dependente da presença de todas as embarcações no local estipulado. A condição inicial de cada embarcação é consequência de sua última campanha, ou conjunto de tarefas realizadas no horizonte de planejamento anterior. Cada embarcação terá encerrada sua campanha em um instante de tempo distinto, e estará à disposição no local de conclusão da última tarefa, aguardando ser programada.

De maneira detalhada, as características principais deste problema são:

Uma operação de apoio marítimo utiliza-se de uma frota de embarcações especiais para atender, dentro de um horizonte de tempo, um conjunto de tarefas possuindo as seguintes características: 1) cada embarcação estará posicionada no local em que encerrar a sua última campanha (conjunto de tarefas realizadas no horizonte de planejamento anterior); 2) cada embarcação estará disponível no instante em que encerrar a sua última campanha; 3) há restrições de precedência entre as tarefas que devem ser respeitadas quando da programação da frota¹⁸; 4) existe uma janela de tempo para início do atendimento de cada tarefa¹⁹; 5) a duração dos roteiros é limitada pela autonomia das embarcações; 6) a frota é heterogênea; 7) embarcações iguais podem ter estruturas de custos diferentes em função das modalidades de afretamento; 8) múltiplas embarcações de tipos diferentes podem ser requisitadas para atender uma tarefa; 9) embarcações compatíveis, embora de outras classes, podem ser utilizadas quando houver falta de uma embarcação específica²⁰; 10) uma requisição de serviço pode iniciar e terminar em locais diferentes²¹. O objetivo é resolver o problema de programação da frota atendendo integralmente à demanda e respeitando as restrições operacionais. A função objetivo do problema é a função de

¹⁸ Por exemplo, o local onde uma plataforma de produção irá operar é primeiramente prospectado por uma plataforma-sonda, tendo os seus poços perfurados. Posteriormente, a unidade marítima será rebocada para este local, posicionada, ancorada e interligada ao sistema produtivo. Programar toda a demanda implica respeitar a ordem em que as tarefas deverão acontecer.

¹⁹ A janela de tempo para este problema consiste de um período pré-definido de dias de que uma tarefa poderá ser antecipada ou atrasada, dentro da macro programação da empresa, sem inviabilizar a operação. Enquanto que a antecipação não onera a produção, o atraso posterga o instante em que uma unidade marítima começará a produzir. Assim, associado ao atraso há uma penalidade diária linear proporcional à perda (ou adiamento) da produção.

²⁰ Uma embarcação de classe superior, isto é, de maior potência, é apta para atender uma requisição que necessita uma embarcação menos potente. A prática adotada é, na falta da embarcação de uma determinada classe, empregar uma embarcação da classe imediatamente superior, pois embarcações ainda mais potentes caracterizariam um mau emprego do recurso.

²¹ As requisições de reboque de uma unidade marítima referem-se ao deslocamento de uma plataforma de perfuração ou de produção de um local para outro. Eventualmente, este deslocamento será a partir de um estaleiro de reparo ou construção para o local de produção, ou de um local em que uma unidade estava realizando um trabalho de prospecção. Daí o início e o término de uma requisição poder referir-se a locais diferentes.

custo total desta operação que engloba as parcelas de custo variável (consumo de combustível das embarcações, ao se reposicionarem de uma tarefa para a seguinte)²² e a penalidade por atraso no cumprimento das datas de cada tarefa (nível de serviço).

3.2 Estratégias de Modelagem do Problema de Programação de Frota

A modelagem matemática seguiu duas abordagens distintas, inspiradas em estratégias diferentes de representação do problema. A primeira estratégia consistiu em utilizar a representação clássica de problemas de “fluxo em rede”. Uma rede orientada $G = (N, A)$, com diversos atributos associados aos seus arcos (custo, distância e tempo) e aos seus nós (demanda por tipo de embarcação, janela de tempo, duração da tarefa e penalidade por atraso), é construída, objetivando-se determinar o fluxo de mínimo custo que atenda à demanda de cada nó da rede, respeitando às restrições operacionais. Uma versão deste modelo, em que as embarcações aptas para atender cada tarefa são agrupadas em conjuntos, também será apresentada.

A segunda estratégia de modelagem é orientada à definição dos instantes de início das tarefas, juntamente com a escolha do conjunto de embarcações para executar cada tarefa. Para isso, criou-se uma escala de tempo discretizada e apenas um dentre os instantes válidos para início da tarefa (instantes dentro da janela de tempo) será escolhido. A discretização consiste em subdividir o período de um dia em frações de dia, como por exemplo, $1/3$ de dia, $1/4$ de dia, etc. Todos os parâmetros de tempo do modelo (durações das tarefas, tempos de viagem, penalidade diária, limites da janela de tempo e instante desejado de início) são ajustados para a unidade de tempo-padrão definida, assim como todas as variáveis relacionadas ao tempo. Três modelos serão apresentados: dois em que são desconsiderados os tempos de

²² Não será considerada a parcela de consumo de combustível das embarcações durante a execução das tarefas. Considera-se que esta parcela é pequena e que não é significativamente alterada com soluções alternativas.

deslocamentos das embarcações do local de término de uma tarefa para o local de início da seguinte, e outra em que os tempos de viagem são incorporados na escala de tempo discreta. Estes modelos serão designados de “modelos de programação de frota em escala de tempo discretizada”.

3.3 Formulação Matemática de Fluxo em Rede

A rede utilizada para a representação do problema é definida como $G=(N,A)$, sendo N o conjunto de nós de rede, e A o conjunto de arcos orientados. Associado aos elementos desta rede (nós e arcos) têm-se os parâmetros que caracterizam a demanda de serviço do problema, que serão, em seguida, apresentados. As características da frota de embarcações que representam a oferta de serviço, também serão apresentadas.

3.3.1 Nós da rede

O conjunto de nós da rede N é dado por $N = N_1 \cup N_2 \cup N_3$. N_1 é uma lista ou coleção ordenada, em que cada elemento $\{0_1, 0_2, \dots, 0_r, \dots, 0_\varepsilon\}$ ²³ representa o nó de origem de cada embarcação r , indicando o local onde cada uma encerrou sua última campanha. Os elementos de N_1 serão referidos pelo índice r , pela notação $N_1(r)$. Cada elemento do conjunto $N_2 = \{1, 2, \dots, \eta\}$ representa uma tarefa que deverá ser atendida no horizonte de planejamento. A lista ou coleção ordenada $N_3 = \{(\eta+1)_1, (\eta+1)_2, \dots, (\eta+1)_r, \dots, (\eta+1)_\varepsilon\}$ representa, para cada embarcação r , sua condição final para o atual horizonte de planejamento, ou seja, o local em que irá encerrar sua última tarefa. Cada elemento de N_3 será referido pelo índice r e pela notação $N_3(r)$. Os elementos do conjunto N serão indexados pelos índices i, j, k e g .

²³ ε é o número total de embarcações da frota, e r o índice usado para o conjunto de embarcações.

3.3.2 Parâmetros referentes às tarefas

Associados aos nós de demanda da rede ($j \in N_2$) são definidos os seguintes parâmetros:

$loc_início(j)$	local de início da tarefa j [coordenadas geodésicas];
$loc_fim(j)$	local de término da tarefa j [coordenadas geodésicas];
$início(j)$	instante desejado para início da tarefa j [dia];
$li(j)$	limite inferior da janela de tempo para início da tarefa j [dia];
$ls(j)$	limite superior da janela de tempo para início da tarefa j [dia];
$penal(j)$	penalidade pelo atraso em relação ao instante desejado no início da tarefa j [\$/dia];
$p(j)$	duração da tarefa j [dia];
$pred(j)$	conjunto de tarefas predecessoras que deverão estar concluídas antes da tarefa j poder iniciar.

3.3.3 Arcos da rede

Os arcos da rede são representados pelo conjunto A , sendo $A = A_1 \cup A_2 \cup A_3 \cup A_4$. O conjunto $A_1 = \{(i, j) : i \in N_1, j \in N_2\}$ é o conjunto de arcos que iniciam no nó de origem de cada embarcação, no início do horizonte de planejamento, e terminam em um nó representativo de uma tarefa. A notação $A_1(r)$ será usada para distinguir os arcos específicos da embarcação $r \in E$, ou seja, indicam os arcos que unem o local da condição inicial de cada embarcação aos nós que representam o local de início de uma tarefa da rede: $A_1(r) = \{(i, j) : i \in N_1(r), j \in N_2\}$. O conjunto $A_2 = \{(i, j) : i \in N_2, j \in N_2, i \neq j, li(i) + p(i) \leq ls(j)\}$ é o conjunto de arcos que iniciam no local de término de cada tarefa i e terminam no local de início de cada tarefa j , isto é, que unem “nós de demanda” da rede. O conjunto A_2 não contempla

os arcos entre as tarefas i e j , que são inviáveis de serem executadas em seqüência, isto é, quando o instante mais cedo de término da tarefa i , dado pela soma do limite inferior da janela de tempo com a duração da tarefa, excede o limite superior da janela de tempo da tarefa j . $A_3 = \{(i, j) : i \in N_2, j \in N_3\}$ define o conjunto de arcos que iniciam nos locais de término das tarefas j e terminam no nó de destino de cada embarcação. Uma vez que cada embarcação permanece no local de término da última tarefa realizada no horizonte de planejamento, os arcos do conjunto A_3 são fictícios e servem para auxiliar a modelagem da condição final da operação das embarcações. A notação $A_3(r)$ será usada para distinguir os arcos específicos da embarcação $r \in E$, ou seja, indicam os arcos que unem o local de conclusão da última tarefa realizada à condição final de cada embarcação: $A_3(r) = \{(i, j) : i \in N_2, j \in N_3(r)\}$. Por último, $A_4 = \{(i, j) : i \in N_1, j \in N_3\}$ contém os arcos ligam a condição inicial à condição final de cada embarcação, também fictícios, e servem para modelar a situação em que uma determinada embarcação não é utilizada na programação. A notação $A_4(r)$ será usada para identificar, para cada embarcação $r \in E$, o arco que une sua condição inicial à condição final: $A_4(r) = \{(i, j) : i \in N_1(r), j \in N_3(r)\}$. Os conjuntos A_1 , A_2 , A_3 , A_4 e A têm dimensão, respectivamente, $m_1 = |A_1|$, $m_2 = |A_2|$, $m_3 = |A_3|$, $m_4 = |A_4|$ e $m = |A| = m_1 + m_2 + m_3 + m_4$.

3.3.4 Parâmetros referentes às embarcações

O modelo matemático deverá determinar o fluxo de embarcações na rede apresentada, de forma a atender às restrições operacionais do problema. Devido à diferenciação que há entre as embarcações do tipo AHTS utilizadas nas atividades de reboque de unidades marítimas e lançamento de linhas de ancoragem, quanto à velocidade, potência de tração estática e capacidade (ou potência) de seus guinchos e, por haver demanda específica de embarcações segregadas por estas características,

faz-se necessário definir classes de embarcações semelhantes, e especificar seus atributos mais relevantes.

Será utilizado o conjunto $C = \{1, 2, \dots, \gamma\}$, indexado pelo índice c , para representar cada uma das γ classes do problema. As embarcações serão representadas pelo conjunto $E = \{1, 2, \dots, \varepsilon\}$, indexado pelo índice r , sendo o ε o número total de embarcações.

O problema desta pesquisa, no caso mais geral, prevê a cessão de embarcações entre classes, ou seja, permite-se o empréstimo de embarcações de uma classe imediatamente superior quando houver falta de embarcações de uma classe desejada; o empréstimo de embarcações de classes superiores à classe imediatamente acima da classe demandada caracterizaria um mau emprego do recurso, devendo ser evitado.

A possibilidade de cessão de embarcações gera a necessidade de definir os índices de quais embarcações são aptas para realizar uma tarefa quando houver demanda por embarcações da classe c . Isto pode ser feito por meio do parâmetro $emb(c)$, em seguida definido:

$emb(c)$ conjunto contendo os índices das embarcações compatíveis com a classe c , isto é, contempla todas as embarcações da classe c , bem como as embarcações da classe subsequente $(c+1)$, quando houver;

Cada embarcação $r \in E$ possui como parâmetros:

$classe(r)$ índice da classe a qual pertence;
 $v(r)$ velocidade da embarcação r [nós];

$cv(r)$	custo variável diário [\$/dia], sendo o produto do custo do combustível [\$/t] pelo consumo diário de combustível [t/dia] na velocidade de serviço (ou, de cruzeiro), da embarcação r ;
$disp(r)$	instante de disponibilidade inicial da embarcação r , em função do contrato de afretamento ou do término da campanha anterior;
$volta(r)$	instante limite de operação da embarcação r , definido no contrato de afretamento;
$autonomia(r)$	autonomia de operação da embarcação r , após o qual ela deverá retornar à base para abastecer e/ou trocar a tripulação.

Cabe ratificar que o custo variável, indicativo do consumo de combustível das embarcações durante seus deslocamentos, não será computado durante a execução das tarefas. Este consumo será considerado apenas nas viagens entre o local de término de uma tarefa e o local de início da tarefa subsequente.

3.3.5 Parâmetros referentes ao binômio tarefa - embarcação

Dois parâmetros referem-se ao binômio tarefa-embarcação:

$q(j,c)$	especifica o número ou a demanda de embarcações da classe $c \in C$ para cada tarefa $j \in N_2$;
$compat(j,r)$	será igual a 1, se a embarcação $r \in E$ for compatível com a tarefa $j \in N_2$; em caso contrário, será igual a 0.

O parâmetro de compatibilidade traz consigo o aspecto da cessão de embarcações entre classes. Assim, uma embarcação r da classe $c+1$ poderá ser compatível com a tarefa j mesmo que a demanda por embarcações da classe $c+1$ seja nula, bastando que a demanda da classe c seja maior do que zero.

Este parâmetro permite contemplar casos em que houver preferência por uma determinada embarcação para executar tarefas específicas. Por exemplo, é possível que determinadas embarcações contenham sistemas propulsores auxiliares, como os “bow thrusters” ou sistemas de manobras acoplados a sistemas de posicionamento dinâmico que facilitem a execução de tarefas consideradas críticas. Assim, pode ser interessante nomear uma determinada embarcação para realizar uma tarefa ao invés de recorrer a qualquer embarcação de sua classe. O uso deste parâmetro permitiria modelar esta situação²⁴.

3.3.6 Duração das viagens

O último parâmetro a ser definido refere-se aos tempos de viagem das embarcações:

$tempo(i,j,r)$ para cada arco $(i,j) \in A$, e para cada embarcação $r \in E$, o parâmetro $tempo(i,j,r)$ expressará o tempo de viagem, em dias, entre a posição de término da tarefa i e a posição de início da tarefa j , dado por $dist(i,j)/24v_r$ ²⁵;

Para os casos em que $i \in N_1$, os tempos de viagem para a embarcação r serão iguais a $+\infty$, se $i \neq N_1(r)$, ou seja, se o nó i se referir à origem de outra embarcação. Se $i = N_1(r)$, então $tempo(i,j,r)$ conterà o tempo de viagem até o local de início das tarefas j do problema. Analogamente, quando $j \in N_3$, o tempo de viagem para a embarcação r será igual a $+\infty$, se $j \neq N_3(r)$, ou seja, se o nó j se referir ao destino de

²⁴ Para os problemas testados nesta pesquisa, considerou-se que não há diferenciação entre embarcações de uma mesma classe para executar uma tarefa de reboque de unidades marítimas e lançamento de linhas de ancoragem.

²⁵ O uso do fator 24 se faz necessário para adequar a unidade de tempo. Sendo a distância dada em milhas e a velocidade expressa em nós (milhas/h), a divisão por 24 converte o tempo de viagem em dias.

outra embarcação. Caso contrário, sendo $j = N_3(r)$, o parâmetro $tempo(i,j,r)$ conterà o tempo de viagem até o local de destino da embarcação, convencionado como sendo o próprio local onde encerrou a última tarefa, portanto, o tempo será nulo.

3.3.7 Modelo de “Fluxo em Rede – Completo”

O modelo matemático, designado de “Fluxo em Rede - Completo” passa a ser definido, nos seguintes termos:

Variáveis de Decisão

$$x_{ij}^r = \begin{cases} 1, & \text{se a embarcação } r \in E \text{ se desloca ao local de início da} \\ & \text{tarefa } j \in N_2 \cup N_3(r) \text{ após ter concluído a tarefa } i \in N_1(r) \cup N_2 \\ 0, & \text{em caso contrário} \end{cases}$$

ini_j - instante de início efetivo da tarefa $j \in N_2$

Variáveis Auxiliares

$Atraso_j$ - tempo de atraso no início de execução da tarefa $j \in N_2$ (medido a partir do instante desejado de início)

ch_{ij}^r - instante de chegada da embarcação $r \in E$ ao local de início da tarefa $j \in N_2 \cup N_3(r)$ após ter concluído a execução da tarefa $i \in N_1(r) \cup N_2$

Função Objetivo

$$\min C = \sum_{j \in N_2} Atraso_j penal(j) + \sum_{r \in E} \sum_{i \in N_1(r) \cup N_2} \sum_{j \in N_2 \cup N_3(r)} (cv(r) x_{ij}^r tempo(i, j, r)) \quad (3.1)$$

Restrições

$$\sum_{j \in N_2 \cup N_3(r)} x_{ij}^r = 1 \quad r \in E, i \in N_1(r) \quad (3.2)$$

$$\sum_{i \in N_1(r) \cup N_2} x_{ij}^r = 1 \quad r \in E, j \in N_3(r) \quad (3.3)$$

$$\sum_{r \in \text{emb}(c)} \sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r \text{compat}(j, r) \geq q(j, c) \quad j \in N_2, c \in C \wedge q(j, c) > 0 \quad (3.4)$$

$$\sum_{r \in E} \sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r \text{compat}(j, r) = \sum_{c \in C} q(j, c) \quad j \in N_2 \quad (3.5)$$

$$\sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r \leq 1 \quad r \in E, j \in N_2 \quad (3.6)$$

$$\sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r \text{compat}(j, r) = \sum_{\substack{k \in N_2 \cup N_3(r) \\ k \neq j}} x_{jk}^r \text{compat}(k, r) \quad r \in E, j \in N_2 \quad (3.7)$$

$$ch_{ij}^r \geq \text{disp}(r) + \text{tempo}(i, j, r) - (1 - x_{ij}^r) T_{\max} \quad r \in E, i \in N_1(r), j \in N_2 \cup N_3(r) \quad (3.8)$$

$$ch_{ij}^r \leq \text{disp}(r) + \text{tempo}(i, j, r) + (1 - x_{ij}^r) T_{\max} \quad r \in E, i \in N_1(r), j \in N_2 \cup N_3(r) \quad (3.9)$$

$$ch_{ij}^r \leq x_{ij}^r T_{\max} \quad r \in E, i \in N_1(r), j \in N_2 \cup N_3(r) \quad (3.10)$$

$$ch_{ij}^r \geq ini_i + p(i) + \text{tempo}(i, j, r) - (1 - x_{ij}^r) T_{\max} \quad r \in E, i \in N_2, j \in N_2 \cup N_3(r), i \neq j \quad (3.11)$$

$$ch_{ij}^r \leq ini_i + p(i) + \text{tempo}(i, j, r) + (1 - x_{ij}^r) T_{\max} \quad r \in E, i \in N_2, j \in N_2 \cup N_3(r), i \neq j \quad (3.12)$$

$$ch_{ij}^r \leq x_{ij}^r T_{\max} \quad r \in E, i \in N_2, j \in N_2 \cup N_3(r), i \neq j \quad (3.13)$$

$$ini_j \geq ch_{ij}^r \quad r \in E, i \in N_1(r) \cup N_2, j \in N_2, i \neq j \quad (3.14)$$

$$ini_j \geq li(j) \quad j \in N_2 \quad (3.15)$$

$$ini_j \leq ls(j) \quad j \in N_2 \quad (3.16)$$

$$ini_j \geq ini_g + p(g) \quad g \in \text{pred}(j), j \in N_2 \quad (3.17)$$

$$\text{Atraso}_j \geq ini_j - \text{inicio}(j) \quad j \in N_2 \quad (3.18)$$

$$ch_{ij}^r \leq \text{volta}(r) \quad r \in E, i \in N_1(r) \cup N_2, j \in N_3(r) \quad (3.19)$$

$$ini_i \geq 0, \text{Atraso}_i \geq 0 \quad i \in N_2 \quad (3.20)$$

$$ch_{ij}^r \geq 0, x_{ij}^r \in \{0, 1\} \quad r \in E, i \in N_1(r) \cup N_2, j \in N_2 \cup N_3(r) \quad (3.21)$$

A equação (3.1) define a função objetivo do problema, a qual é composta das parcelas de atraso ponderado das tarefas e o custo variável advindo do

reposicionamento das embarcações. A restrição (3.2) garante que cada embarcação r inicia no local de sua condição inicial, dirigindo-se para os nós dos conjuntos N_2 ou $N_3(r)$. A restrição (3.3) impõe a condição final de cada embarcação. A restrição (3.4) impõe que o número de embarcações alocadas por classe a uma tarefa seja maior ou igual ao mínimo requerido por aquela classe. Esta restrição só será aplicada se houver demanda de embarcações da classe c ; a somatória incidirá sobre todas as embarcações do conjunto $emb(c)$, isto é, as embarcações compatíveis com a classe c . A restrição (3.5) complementa a restrição (3.4), forçando que o total de embarcações que executam uma tarefa seja igual ao total demandado. Esta restrição é necessária por causa da possibilidade de cessão de embarcações entre classes, de forma a evitar que uma mesma embarcação seja contada mais de uma vez, para uma mesma tarefa. A restrição (3.6) impede que uma embarcação r seja alocada a mais de uma tarefa a partir de um mesmo nó de origem. A restrição (3.7) garante que as embarcações que deixam um local tenham necessariamente chegado lá, ou seja, impõe a continuidade do fluxo. O par de restrições (3.8) e (3.9) define o instante de chegada de uma embarcação ao local de início de uma tarefa, considerando a disponibilidade inicial desta embarcação e o tempo necessário de deslocamento. A constante $T_{\max}$ pode ser determinada, fixando-se o seu valor igual à duração do horizonte de planejamento do problema. A restrição (3.10) zera o valor de ch_{ij}^r quando não houver fluxo da embarcação r de i para j . As restrições (3.11) e (3.12) desempenham um papel similar ao das restrições (3.8) e (3.9) fixando o instante de chegada da embarcação r no local de início da tarefa j após ter executado a tarefa i sendo, portanto, dependente do instante de início e da duração da tarefa i , bem como do tempo de deslocamento entre i e j . A restrição (3.13) é equivalente à restrição (3.10) e garante que a variável ch_{ij}^r será nula quando não houver fluxo da embarcação r de i para j . A restrição (3.14) impõe que não poderá haver início da tarefa j enquanto todas as embarcações a ela designadas não tiverem chegado ao local de início dessa tarefa. A restrição (3.15) impede que haja início da tarefa j antes do limite inferior da janela de tempo. O limite

superior da janela de tempo para início da tarefa j é garantido pela restrição (3.16). A restrição (3.17) trata da imposição de precedência entre tarefas, forçando que todas as tarefas predecessoras g estejam concluídas para que a tarefa j possa iniciar. A restrição (3.18) auxilia no cálculo do atraso para o início da tarefa j . O retorno da embarcação à base até a data limite estabelecida no contrato de afretamento é garantido pela restrição (3.19). Por último as restrições (3.21) impõem a não negatividade das variáveis ini_j , $Atraso_j$ e ch_{ij}^r , e garantem que x_{ij}^r seja binária.

O modelo apresentado contém em sua estrutura algumas restrições redundantes que podem ser suprimidas sem alterar o conjunto de soluções viáveis. Entre elas está a restrição (3.6), que garante que uma embarcação r chega a um determinado nó j vinda de apenas uma origem. Contudo, considerando que a restrição (3.2) obriga cada embarcação a sair da condição inicial e, ao chegar em um determinado nó, a restrição (3.7) impõe a continuidade do fluxo, e sendo as variáveis binárias, no máximo um nó receberá a visita subsequente desta embarcação. As restrições (3.9), (3.10), (3.12) e (3.13) também são dispensáveis, considerando que a variável que registra o instante de chegada ch_{ij}^r pode assumir qualquer valor, desde que respeite o instante de início da viagem (disponibilidade inicial da embarcação ou término de uma tarefa), a existência do fluxo e a duração dos tempos de viagem.

3.3.8 Modelo de “Fluxo em Rede – Compacto”

O conjunto simplificado de restrições passa a ter estrutura abaixo, dando origem ao modelo designado de “Fluxo em Rede – Compacto”.

Restrições

$$\sum_{j \in N_2 \cup N_3(r)} x_{ij}^r = 1 \quad r \in E, i \in N_1(r) \quad (3.21)$$

$$\sum_{i \in N_1(r) \cup N_2} x_{ij}^r = 1 \quad r \in E, j \in N_3(r) \quad (3.22)$$

$$\sum_{r \in \text{emb}(c)} \sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r \text{compat}(j, r) \geq q(j, c) \quad j \in N_2, c \in C \wedge q(j, c) > 0 \quad (3.23)$$

$$\sum_{r \in E} \sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r \text{compat}(j, r) = \sum_{c \in C} q(j, c) \quad j \in N_2 \quad (3.24)$$

$$\sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r \text{compat}(j, r) = \sum_{\substack{k \in N_2 \cup N_3(r) \\ k \neq j}} x_{jk}^r \text{compat}(k, r) \quad r \in E, j \in N_2 \quad (3.25)$$

$$ch_{ij}^r \geq \text{disp}(r) + \text{tempo}(i, j, r) - (1 - x_{ij}^r) T_{\max} \mid \quad r \in E, i \in N_1(r), j \in N_2 \cup N_3(r) \quad (3.26)$$

$$ch_{ij}^r \geq \text{ini}_i + p(i) + \text{tempo}(i, j, r) - (1 - x_{ij}^r) T_{\max} \mid \quad r \in E, i \in N_2, j \in N_2 \cup N_3(r), i \neq j \quad (3.27)$$

$$\text{ini}_j \geq ch_{ij}^r \quad r \in E, i \in N_1(r) \cup N_2, j \in N_2, i \neq j \quad (3.28)$$

$$\text{ini}_j \geq \text{li}(j) \quad j \in N_2 \quad (3.29)$$

$$\text{ini}_j \leq \text{ls}(j) \quad j \in N_2 \quad (3.30)$$

$$\text{ini}_j \geq \text{ini}_g + p(g) \quad g \in \text{pred}(j), j \in N_2 \quad (3.31)$$

$$\text{Atraso}_j \geq \text{ini}_j - \text{inicio}(j) \quad j \in N_2 \quad (3.32)$$

$$ch_{ij}^r \leq \text{volta}(r) \quad r \in E, i \in N_1(r) \cup N_2, j \in N_3(r) \quad (3.33)$$

$$\text{ini}_i \geq 0, \text{Atraso}_i \geq 0 \quad i \in N_2 \quad (3.34)$$

$$ch_{ij}^r \geq 0, x_{ij}^r \in \{0, 1\} \quad r \in E, i \in N_1(r) \cup N_2, j \in N_2 \cup N_3(r) \quad (3.35)$$

A modelagem de fluxo em rede apresentada é uma variação do modelo clássico encontrado na literatura (Laporte; Osman, 1995), tendo sido acrescentadas as restrições (3.23) e (3.24) de requisição de múltiplas embarcações. Em geral, a restrição relativa à demanda, nos modelos clássicos de roteirização e programação de veículos, obriga a soma das variáveis do fluxo incidente em cada nó ser igual a um, dado que a demanda é atendida por um único veículo. Caso não houvesse a política de cessão de embarcações entre classes, as restrições (3.23) e (3.24) poderiam ser substituídas pela restrição (3.23'), em que a demanda por classe é garantida:

$$\sum_{r \in \text{emb}(c)} \sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r \text{compat}(j, r) = q(j, c) \quad j \in N_2, c \in C \wedge q(j, c) > 0 \quad (3.23')$$

A restrição (3.28) impede que o início de uma tarefa ocorra antes da chegada de todas as embarcações a ela designadas, garantindo assim, a simultaneidade das operações.

3.3.9 Extensões do Modelo

Os modelos apresentados podem incorporar aspectos complementares à operação de lançamento de linhas de ancoragem e reboque de plataformas de petróleo. Um destes aspectos é a possibilidade de afretamento de novas embarcações em períodos de alta demanda; em tal situação, o modelo permitiria avaliar o balanço (“trade off”) entre a redução das penalidades por atraso no início das tarefas com os custos decorrentes das novas embarcações incorporadas à frota.

O retorno da embarcação à base para troca da tripulação e abastecimento também será modelado. Considerando que a troca de tripulação ocorre na base, é importante contemplar o retorno das embarcações de forma a aproximar os resultados à realidade.

Por último, como extensão do modelo de fluxo em rede, será apresentada a formulação com conjuntos de embarcações, em que as embarcações aptas para atender uma tarefa são agrupadas previamente em conjuntos – e o modelo seleciona um conjunto viável para cada tarefa.

3.3.9.1 Afretamento

Embora a ênfase desta pesquisa esteja voltada para a questão de programação da frota, é possível incorporar decisões de afretamento de novas embarcações, estendendo o escopo do problema abordado nesta pesquisa.

Para isso, é necessário modificar o conjunto E de embarcações para incluir além das que estão ativas, as embarcações candidatas a fazerem parte da frota. A diferenciação sobre qual embarcação já está afretada poderá ser explicitada pelo parâmetro:

$$ativa(r) \begin{cases} 1, \text{ se a embarcação } r \text{ pertencer à frota} \\ 0, \text{ em caso contrário} \end{cases}$$

Além disso, é necessário informar o custo de afretamento para cada embarcação r :

$$afret(r) \quad \text{zero para as embarcações } r \text{ que pertencem à frota e, para as} \\ \text{embarcações candidatas, igual ao produto da taxa diária de afretamento} \\ \text{pelo período de afretamento}^{26}.$$

Uma nova variável de decisão é introduzida para escolher quais embarcações irão compor a frota:

$$w_r = \begin{cases} 1, \text{ se a embarcação } r \in E \text{ for utilizada} \\ 0, \text{ em caso contrário} \end{cases}$$

²⁶ Este período pode ter sua extensão compatível com um eventual pico de demanda para o qual a embarcação esteja sendo afretada ou, pode ser um período maior, em que a taxa de afretamento seja mais atrativa, de acordo as condições do mercado de afretamento. Caso a extensão do contrato seja significativamente maior que o horizonte de planejamento, será necessário ajustar o parâmetro representativo do período no modelo matemático, de forma a não inviabilizar o afretamento.

Cabe observar que sendo r uma embarcação já afretada, isto é, que já faz parte da frota, automaticamente w_r será igual a 1.

Dessa maneira, a função objetivo fica alterada para (3.36):

$$\min C = \sum_{j \in N_2} Atraso_j penal(j) + \sum_{r \in E} \sum_{i \in N_1(r) \cup N_2} \sum_{j \in N_2 \cup N_3(r)} (x_{ij}^r cv(r) tempo(i, j, r)) + \sum_{r \in E} w_r afret(r) \quad (3.36)$$

Para permitir a seleção e a incorporação das embarcações à frota são acrescentadas as restrições (3.37), (3.38) e (3.39), abaixo apresentadas.

$$\sum_{i \in N_1(r)} \sum_{j \in N_2} x_{ij}^r \leq w_r \quad r \in E \quad (3.37)$$

$$w_r \geq ativa(r) \quad r \in E \quad (3.38)$$

$$w_r \in \{0,1\} \quad r \in E \quad (3.39)$$

A restrição (3.37) garante que, se a embarcação r não estiver ativa, nenhum fluxo será permitido para os nós de demanda. Neste caso, para que a restrição (3.21), que obriga todas as embarcações saírem da condição inicial seja respeitada, o único fluxo possível será para o nó de destino $(\eta+1)_r$, a um custo nulo, dado que este arco é fictício. Caso a embarcação seja ativada no modelo, esta poderá realizar, inicialmente, no máximo 1 tarefa, e pela restrição de continuidade (3.25), a embarcação poderá ser designada a outras tarefas. A restrição (3.38) garante que as embarcações que já estão ativas, isto é, que pertencem à frota, sejam automaticamente habilitadas. Por último, a restrição (3.39) impõe que as variáveis sejam binárias.

3.3.9.2 *Abastecimento & Troca de Tripulação*

A aderência do modelo à realidade requer que seja considerado o retorno das embarcações à base para troca de tripulação e abastecimento. O intervalo médio, em dias, para retorno das embarcações é dado pelo parâmetro $autonomia(r)$. Considera-se que, uma embarcação, ao retornar à base para troca de tripulação, poderá também ser abastecida. Caso esta premissa não seja válida, poderão ser empregados dois parâmetros distintos para especificar a duração dos roteiros: um referente à periodicidade de troca de tripulação e outro relativo à autonomia.

Para representar o retorno das embarcações para trocar tripulação e para abastecer, serão criadas tarefas específicas de retorno à base, que somente poderão ser realizadas pelas embarcações correspondentes. Associada a cada tarefa haverá uma janela de tempo restrita. Esta janela rígida, em parte, é devido à disponibilidade de berço para atracação e para que as leis trabalhistas sejam respeitadas, isto é, para que o cronograma de troca de tripulação seja rigorosamente cumprido.

Cabe observar que as datas destas tarefas dependem do cronograma de troca de tripulação e abastecimento realizado no horizonte de planejamento anterior, devendo ser mantido o intervalo médio de retorno à base.

No modelo, admite-se que qualquer embarcação de uma classe está apta a executar uma tarefa (se a classe for compatível com aquela tarefa), podendo ainda haver empréstimo de embarcações entre classes. Para evitar que seja designada uma embarcação diferente à tarefa de abastecimento, é necessário ajustar o parâmetro de compatibilidade – $compat(j,r)$, de forma que, apenas a embarcação r possa realizar a sua respectiva tarefa de abastecimento j . Portanto, as tarefas específicas de abastecimento terão o parâmetro $compat(j,r)=1$ para cada embarcação r designada a ser abastecida, e zero para as demais embarcações.

3.3.9.3 *Agrupamento em Conjunto de Embarcações*

Observou-se ser conveniente, quando da elaboração dos métodos de solução empregados nesta pesquisa, o agrupamento das embarcações candidatas à execução de uma dada tarefa em conjuntos. Como múltiplas embarcações podem ser requisitadas para uma tarefa, qualquer agrupamento atendendo à especificação da demanda de embarcações por classe, configura-se um conjunto candidato de embarcações para uma tarefa.

A introdução do agrupamento em conjuntos, previamente definidos, torna a implementação dos métodos heurísticos mais simples. Como há a possibilidade de cessão de embarcações entre classes, ao se analisar, por exemplo, uma fila de tarefas aguardando alocação e uma lista de embarcações livres, seria ineficiente, do ponto de vista computacional, checar se as embarcações livres compõem um conjunto viável, capaz de atender a alguma tarefa da fila. O agrupamento em conjuntos permite monitorar o instante em que cada conjunto é capaz de iniciar uma tarefa da fila, tendo em vista os instantes de término das tarefas nas quais as embarcações do conjunto estão operando e os tempos necessários para se deslocarem até o local de início da nova tarefa. Desta forma, não é necessário monitorar quando as embarcações ficam livres, antes o que deve ser acompanhado é o status dos conjuntos, quando estes ficam livres e os instantes em que conseguiriam iniciar a execução das tarefas em fila.

A implementação utilizando a estrutura de conjuntos requer a reestruturação do modelo matemático. Para isso, foi necessário incorporar dois parâmetros adicionais:

$Conj = [conj_{h,r}]$: uma matriz com dimensão $\phi \times \varepsilon$, onde ϕ é o número de todos os possíveis conjuntos para as η tarefas do problema; ε é a quantidade de embarcações. Se o conjunto h contiver a embarcação r , então o parâmetro $[conj_{h,r}]$ será igual a 1 e igual a 0, em caso contrário;

$conj_tar(j)$ contém os índices h dos conjuntos de embarcações candidatas para realizar a tarefa j .

Os conjuntos são construídos por meio de uma rotina que enumera todas as possíveis combinações de embarcações, considerando a cessão entre classes. Por exemplo: i) supondo haver 2 classes de embarcações, contendo cada classe, respectivamente, 2 e 3 embarcações; ii) supondo que a demanda de uma determinada tarefa seja (2;1), isto é, duas embarcações da primeira classe e uma da segunda, a tabela 3.1 indica as possíveis combinações para atender a demanda desta tarefa.

Tabela 3.1 – Exemplo de Conjuntos de Embarcações

Classe	1		2		
Embarcação	1	2	3	4	5
Conjunto 1	■	■	■		
Conjunto 2	■	■		■	
Conjunto 3	■	■			■
Conjunto 4	■		■	■	
Conjunto 5	■		■		■
Conjunto 6	■			■	■
Conjunto 7		■	■	■	
Conjunto 8		■	■		■
Conjunto 9		■		■	■
Conjunto 10			■	■	■

Além desses parâmetros, será empregada uma nova variável de decisão z_h que decidirá pela escolha de um conjunto h de embarcações para realizar a tarefa j , dentre os possíveis conjuntos identificados pelo parâmetro $conj_tar(j)$.

O novo modelo passa a ser apresentado, sendo designado de “Fluxo em Rede - Conjuntos”:

Variáveis de Decisão

$$x_{ij}^r = \begin{cases} 1, & \text{se a embarcação } r \in E \text{ se desloca ao local de início da} \\ & \text{tarefa } j \in N_2 \cup N_3(r) \text{ após ter concluído a tarefa } i \in N_1(r) \cup N_2 \\ 0, & \text{em caso contrário} \end{cases}$$

ini_j - instante de início efetivo da tarefa $j \in N_2$

$$z_h = \begin{cases} 1, & \text{se o conjunto de embarcações } h \ (1 \leq h \leq \phi) \text{ for selecionado} \\ 0, & \text{em caso contrário} \end{cases}$$

Variáveis Auxiliares

$Atraso_j$ - tempo de atraso no início de execução da tarefa $j \in N_2$

ch_{ij}^r - instante de chegada da embarcação $r \in E$ ao local de início da tarefa $j \in N_2 \cup N_3(r)$ após ter concluído a execução da tarefa $i \in N_1(r) \cup N_2$

Função Objetivo

$$\min C = \sum_{j \in N_2} Atraso_j penal(j) + \sum_{r \in E} \sum_{i \in N_1(r) \cup N_2} \sum_{j \in N_2 \cup N_3(r)} (x_{ij}^r cv(r) tempo(i, j, r)) \quad (3.40)$$

Restrições

$$\sum_{j \in N_2 \cup N_3(r)} x_{ij}^r = 1 \quad r \in E, i \in N_1(r) \quad (3.41)$$

$$\sum_{i \in N_1(r) \cup N_2} x_{ij}^r = 1 \quad r \in E, j \in N_3(r) \quad (3.42)$$

$$\sum_{r \in E} \sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r compat(j, r) = \sum_{c \in C} q(j, c) \quad j \in N_2 \quad (3.43)$$

$$\sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r compat(j, r) = \sum_{\substack{k \in N_2 \cup N_3(r) \\ k \neq j}} x_{jk}^r compat(k, r) \quad r \in E, j \in N_2 \quad (3.44)$$

$$ch_{ij}^r \geq (disp(r) + tempo(i, j, r)) - (1 - x_{ij}^r) T_{\max} \quad r \in E, i \in N_1(r), j \in N_2 \cup N_3(r) \quad (3.45)$$

$$ch_{ij}^r \geq (ini_i + p(i) + tempo(i, j, r)) - (1 - x_{ij}^r) T_{\max} | \quad r \in E, i \in N_2, j \in N_2 \cup N_3(r), i \neq j \quad (3.46)$$

$$ini_j \geq ch_{ij}^r \quad r \in E, i \in N_1(r) \cup N_2, j \in N_2, i \neq j \quad (3.47)$$

$$ini_j \geq li(j) \quad j \in N_2 \quad (3.48)$$

$$ini_j \leq ls(j) \quad j \in N_2 \quad (3.49)$$

$$ini_j \geq ini_g + p(g) \quad g \in pred(j), j \in N_2 \quad (3.50)$$

$$Atraso_j \geq ini_j - inicio(j) \quad j \in N_2 \quad (3.51)$$

$$ch_{ij}^r \leq volta(r) \quad r \in E, i \in N_1(r) \cup N_2, j \in N_3(r) \quad (3.52)$$

$$ini_i \geq 0, Atraso_i \geq 0 \quad i \in N_2 \quad (3.53)$$

$$ch_{ij}^r \geq 0, x_{ij}^r \in \{0,1\} \quad r \in E, i \in N_1(r) \cup N_2, j \in N_2 \cup N_3(r) \quad (3.54)$$

$$\sum_{h \in conj_tar(j)} z_h = 1 \quad j \in N_2 \quad (3.55)$$

$$x_{ij}^r \leq \sum_{h \in conj_tar(j)} z_h conj(h, r) \quad r \in E, i \in N_1(r) \cup N_2, j \in N_2, i \neq j \quad (3.56)$$

$$z_h \in \{0,1\} \quad 1 \leq h \leq \phi \quad (3.57)$$

Comparando com a formulação compacta de fluxo em rede, anteriormente apresentada, cabe mencionar que a restrição (3.23) foi suprimida, dando lugar às restrições (3.55), (3.56) e (3.57). A restrição (3.55) impõe que um conjunto dentre os possíveis deverá ser necessariamente escolhido. A restrição (3.56) garante que o fluxo x_{ij}^r será zerado se a embarcação r não pertencer ao conjunto h da tarefa j e/ou se o conjunto h não for designado à tarefa j . Por último, a restrição (3.57) força que as variáveis z_h sejam binárias.

3.4 Modelo de Programação de Frota em Escala de Tempo Discretizada

Para o problema desta pesquisa, foi desenvolvido um segundo tipo de modelagem visando determinar a programação da frota, por meio da definição dos instantes de início de cada tarefa juntamente com a escolha das embarcações. Para isso, utilizou-se como analogia o processo de construção do gráfico de Gantt de um projeto, em que cada atividade é programada (isto é, seu instante de início é definido) alocando um conjunto de recursos produtivos durante o período previsto de execução. Neste caso, a designação dos recursos tem que respeitar a disponibilidade temporal dos mesmos, ou seja, não se pode alocar mais recursos do que os disponíveis.

Esta abordagem é inspirada na forma como os programadores inicialmente alocam a frota. As tarefas que compõem a demanda, aguardando serem programadas, devem iniciar dentro de suas janelas de tempo. Para isso, os programadores buscam distribuir as tarefas, designando-as às embarcações que estiverem livres. Todas as restrições operacionais devem ser observadas, como, por exemplo, as restrições de demanda de embarcações por classe, a simultaneidade no uso das embarcações e as restrições de precedência. A alocação deve ser orientada pela priorização das tarefas mais críticas (as de maior penalidade), buscando, simultaneamente, a minimização dos custos variáveis.

O aspecto fundamental do modelo para esta abordagem é a discretização do tempo. Isto significa que os instantes de início e término das tarefas, bem com os tempos de viagens das embarcações, não serão representados por variáveis reais. A modificação introduzida utiliza uma unidade padrão de tempo, por exemplo, $\frac{1}{2}$ dia ou $\frac{1}{4}$ dia. Assim, todos os parâmetros associados ao tempo (instante de início das tarefas, limite inferior e superior da janela de tempo, duração das tarefas,

penalidades “diárias”, custo variável “diário” e tempos de viagem) passam a ser expressos na unidade padrão de tempo.

Os parâmetros utilizados nesta modelagem são iguais aos apresentados na seção 3.1, com a ressalva da adaptação para a unidade padrão de tempo, quando for pertinente. Utiliza-se, também, o parâmetro T_{MAX} , indicando o horizonte de planejamento.

Um parâmetro adicional será introduzido para indicar se uma dada embarcação r , $r \in E$, estará disponível num dado instante t , $t:1..T_{MAX}$. Considerando que estas têm instantes de disponibilidade inicial distintos, em função do término de suas campanhas anteriores, e que poderão ter que retornar à base caso seus contratos de afretamento encerrem, será utilizada a matriz $A=[a_{t,r}]$, $t:1..T_{MAX}$, $r \in E$ para representar estes aspectos. Cada elemento $[a_{t,r}]$ será igual a 1, se a embarcação r estiver afretada e disponível no instante t , e 0, caso contrário.

Três modelos serão apresentados. Dois deles não contemplam os tempos de deslocamento e consideram que, após o fim de uma tarefa, as embarcações possam iniciar imediatamente o atendimento de outra tarefa. Um terceiro modelo incorpora os tempos de viagem das embarcações. O conjunto N de tarefas, definido anteriormente na formulação de fluxo em rede, não conterá as tarefas fictícias $\{0\}$ e $\{\eta+1\}$ para os dois primeiros modelos, já que eles não incorporam os tempos de deslocamento das embarcações.

3.4.1 *Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão I*

O primeiro modelo, designado de “Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão I’’, passa a ser apresentado.

Variáveis de Decisão

$$z_j^t = \begin{cases} 1, & \text{se a tarefa } j \in N \text{ tiver início no instante } 1 \leq t \leq T_{MAX} \\ 0, & \text{em caso contrário} \end{cases}$$

$$x_{rj} = \begin{cases} 1, & \text{se a tarefa } j \in N \text{ utilizar a embarcação } r \in E \\ 0, & \text{em caso contrário} \end{cases}$$

Variáveis Auxiliares

$$y_{rj}^t = \begin{cases} 1, & \text{se a embarcação } r \in E \text{ estiver alocada à tarefa } j \in N \\ & \text{no instante } t, 1 \leq t \leq T_{MAX} \\ 0, & \text{em caso contrário} \end{cases}$$

$Atraso_j$ - atraso no início da tarefa $j \in N$

ini_j - instante de início efetivo da tarefa $j \in N$

Função Objetivo

$$\min \text{CustoTotal} = \sum_{j \in N} \text{Atraso}_j \text{penal}(j) \quad (3.58)$$

Restrições

$$\sum_{t=li(j)}^{ls(j)} z_j^t = 1 \quad j \in N \quad (3.59)$$

$$\sum_{t=li(j)}^{ls(j)} z_j^t \geq \sum_{t=1}^{T_{MAX}} z_h^t (t + p(h)) \quad h \in \text{pred}(j), j \in N \quad (3.60)$$

$$\sum_{r \in \text{emb}(c)} x_{rj} \text{compat}(j, r) \geq q(j, c) \quad j \in N, c \in C \wedge q(j, c) > 0 \quad (3.61)$$

$$\sum_{r \in E} x_{rj} \text{compat}(j, r) = \sum_{c \in C} q(j, c) \quad j \in N \quad (3.62)$$

$$\sum_{t=li(j)}^{ls(j)+p(j)-1} y_{rj}^t \leq x_{rj} \text{compat}(j,r)p(j) \quad j \in N, r \in E \quad (3.63)$$

$$\sum_{s=t}^{t+p(j)-1} a_{sr} y_{rj}^s \geq x_{rj} \text{compat}(j,r)p(j) - (1 - z_j^t)p(j) \quad j \in N, r \in E, t: li(j), \dots, ls(j) \quad (3.64)$$

$$\sum_{j \in N} y_{rj}^t \leq 1 \quad r \in E, t: 1, \dots, T_{MAX} \quad (3.65)$$

$$ini_j = \sum_{t=li(j)}^{ls(j)} t z_j^t \quad j \in N \quad (3.66)$$

$$Atraso_j \geq ini_j - inicio(j) \quad j \in N \quad (3.67)$$

$$Atraso_j \geq 0, ini_j \geq 0 \quad j \in N \quad (3.68)$$

$$z_j^t \in \{0,1\} \quad j \in N, t: 1, \dots, T_{MAX} \quad (3.69)$$

$$x_{rj} \in \{0,1\} \quad j \in N, r \in E \quad (3.70)$$

$$y_{rj}^t \in \{0,1\} \quad j \in N, r \in E, t: 1, \dots, T_{MAX} \quad (3.71)$$

A eq.(3.58) é a função que expressa o objetivo do problema, ou seja, a minimização das penalidades por atraso no início das tarefas. A restrição (3.59) garante que toda a demanda será atendida, impondo que o instante de início de cada tarefa esteja contido na correspondente janela de tempo. A restrição (3.60) evita que uma tarefa j inicie antes que todas as tarefas predecessoras h sejam finalizadas. A restrição (3.61) visa atender ao número de embarcações solicitadas de cada classe c . Como uma classe superior poderá ceder embarcações para a classe imediatamente inferior, uma mesma embarcação corre o risco de ser considerada alocada por duas classes adjacentes. Para evitar este problema, a restrição (3.62) impõe que o total de embarcações alocadas deverá ser igual à somatória de embarcações requisitadas por todas as classes. Para isso, apenas as embarcações compatíveis com a tarefa são consideradas. A restrição (3.63) garante que, se uma embarcação r for escolhida para executar uma tarefa j (por meio da variável x_{rj}), então a variável que expressa a alocação de cada embarcação ao longo do tempo, y_{rj}^t , assumirá no máximo p_j valores iguais a um (p_j é a duração da tarefa j). Se a embarcação r não for escolhida, esta

restrição fará com que todo $y_{rj}^t = 0$. A restrição (3.64) garante que todas as embarcações r designadas para a realização da tarefa j iniciarão os serviços no mesmo instante t , em que $z_j^t = 1$ e estarão alocadas durante p_j períodos. A restrição (3.65) impede que uma embarcação esteja alocada a mais de uma tarefa em um mesmo instante t . A restrição (3.66) define o instante de início efetivo de cada tarefa j , em função da variável z_j^t . A restrição (3.67) auxilia no cálculo do atraso para início de uma tarefa. Por último, as restrições (3.68) a (3.71) garantem as condições de integralidade e não-negatividade das variáveis.

3.4.2 Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão II

O segundo modelo, designado de “Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão II” difere do modelo I com relação ao aspecto de garantia de simultaneidade na alocação das embarcações.

Para forçar que todas as embarcações iniciem e terminem juntas, serão utilizados três parâmetros:

- $M = [m_{g,t}]$ uma matriz com dimensão $\psi \times T_{MAX}$, onde ψ é o número de todos os possíveis instantes de início das η tarefas: $\psi = \sum_{j \in N} (ls(j) - li(j) + 1)$. Cada elemento $[m_{g,t}]$ será igual 1 se, no instante t , a tarefa representada pelo índice g estiver em operação, dado que o seu início ocorreu em $inst(g)^{27}$;
- $conj_ini(j)$ contém os índices g dos possíveis instantes de início efetivo de cada tarefa j ;
- $inst(g)$ indica o instante de tempo de início associado ao índice g .

²⁷ Ver exemplo na página 92.

A matriz M conterá, para cada tarefa j , um número de linhas proporcional ao número de possibilidades de início da tarefa, dado por $ls(j)-li(j)+1$, considerando a discretização do tempo utilizada. Para que estas ocorrências sejam facilmente identificáveis, as linhas da matriz M serão ordenadas por tarefa, e o $conj_ini(j)$ conterá os índices referentes às linhas da matriz M que contém as informações da tarefa j .

O segundo modelo passa a ser definido nos seguintes termos:

Variáveis de Decisão

$$w_j^g = \begin{cases} 1, & \text{se a tarefa } j \in N \text{ tiver início no instante } inst(g), g \in conj_ini(j) \\ 0, & \text{em caso contrário} \end{cases}$$

$$x_{rj} = \begin{cases} 1, & \text{se a tarefa } j \in N \text{ utilizar a embarcação } r \in E \\ 0, & \text{em caso contrário} \end{cases}$$

Variáveis Auxiliares

$Atraso_j$ - atraso no início da tarefa $j \in N$

$$y_{rj}^t = \begin{cases} 1, & \text{se a embarcação } r \in E \text{ estiver alocada à tarefa } j \in N \\ & \text{no instante } t, 1 \leq t \leq T_{MAX} \\ 0, & \text{caso contrário} \end{cases}$$

ini_j - instante de início efetivo da tarefa $j \in N$

Função Objetivo

$$\min CustosTotal = \sum_{j \in N} Atraso_j penal(j) \quad (3.72)$$

Restrições

$$\sum_{g \in conj_ini(j)} w_j^g = 1 \quad j \in N \quad (3.73)$$

$$\sum_{g \in conj_ini(j)} inst(g) w_j^g \geq \sum_{h \in conj_ini(i)} inst(h) w_i^h + p(i) \quad i \in pred(j), j \in N \quad (3.74)$$

$$\sum_{r \in emb(c)} x_{rj} compat(j, r) \geq q(j, c) \quad j \in N, c \in C \wedge q(j, c) > 0 \quad (3.75)$$

$$\sum_{r \in E} x_{rj} \text{compat}(j, r) = \sum_{c \in C} q(j, c) \quad j \in N \quad (3.76)$$

$$a(t, r) y_{rj}^t \geq \left(\sum_{g \in \text{conj_ini}(j)} m(g, t) w_j^g \right) + x_{rj} - 1 \quad j \in N, r \in E, t: li(j), \dots, ls(j) + p(j) - 1 \quad (3.77)$$

$$\sum_{j \in N} y_{rj}^t \leq 1 \quad r \in E, t: 1, \dots, T_{MAX} \quad (3.78)$$

$$ini_j = \sum_{g \in \text{conj_ini}(j)} inst(g) w_j^g \quad j \in N \quad (3.79)$$

$$Atraso_j \geq ini_j - inicio(j) \quad j \in N \quad (3.80)$$

$$Atraso_j \geq 0, ini_j \geq 0 \quad j \in N \quad (3.81)$$

$$x_{rj} \in \{0, 1\} \quad j \in N, r \in E \quad (3.82)$$

$$w_j^g \in \{0, 1\} \quad j \in N, 1 \leq g \leq \psi \quad (3.83)$$

$$y_{rj}^t \geq 0 \quad j \in N, r \in E, t: 1, \dots, T_{MAX} \quad (3.84)$$

A eq.(3.72) é a função que expressa o objetivo do problema, ou seja, a minimização custo associado ao nível de serviço. A restrição (3.73) impõe que um período de início, dentro da faixa que constitui a janela de tempo, deverá ser necessariamente escolhido para cada tarefa j . A restrição (3.74) faz com que o início de qualquer tarefa j apenas ocorra quando as tarefas predecessoras tiverem sido concluídas. A restrição (3.75) garante que a demanda por classe de embarcação seja atendida. Como é possível haver a cessão de embarcações entre classes, a restrição (3.76) impede que uma embarcação seja computada mais de uma vez, garantindo o atendimento da demanda quanto à totalidade de embarcações requeridas. A restrição (3.77) faz com que a variável y_{rj}^t assumo o valor 1 se a embarcação r iniciou a tarefa ($x_{rj} = 1$) em qualquer $inst(g)$ tal que $[m_{g,t}] = 1$. Contudo, se a embarcação r não estiver afretada (disponível) em algum instante t ($[a_{t,r}] = 0$), então necessariamente a embarcação não poderá ser designada a esta tarefa ($x_{rj} = 0$). A restrição (3.78) impede que uma embarcação esteja alocada a mais de uma tarefa ao mesmo tempo. A restrição (3.79) define o instante efetivo de início da tarefa j , em função da variável

de decisão w_j^g . A restrição (3.80) calcula o atraso para início da tarefa, em relação ao instante desejado, dado pelo parâmetro $início(j)$. Por último, as restrições (3.81) a (3.84) impõem a não-negatividade e a integralidade das variáveis de decisão.

Comparando com a versão I, observa-se que as restrições (3.63) e (3.64) foram substituídas pela restrição (3.77). Assim, ao invés de ter que garantir que no máximo p_j elementos de y_{rj}^t assumam valores iguais a 1, quando $x_{rj}=1$ (3.63) e que o instante de tempo de início seja igual para todas as embarcações selecionadas (3.64), a restrição (3.77) faz uso da matriz M , a qual contém o período de alocação das embarcações para cada opção de início.

Como ilustração, considerar que uma determinada tarefa pode iniciar em qualquer momento entre os instantes 4 e 9, e que sua duração corresponde a 7 períodos, conforme mostra a tabela 3.2. A matriz M contém, para cada opção de início, a indicação do período em que as embarcações deverão estar alocadas, fazendo (parcialmente) o papel das restrições (3.63) e (3.64). Na versão II, a restrição (3.77) é utilizada apenas para atualizar o valor de y_{rj}^t , caso o início da tarefa j tenha ocorrido no período g , e a embarcação r tenha sido designada à tarefa j . Em seguida, a restrição (3.78) impedirá que uma mesma embarcação esteja alocada simultaneamente a mais de uma tarefa.

Tabela 3.2 – Exemplo de Utilização da Matriz M

	Possíveis Instantes de Início - Escala de Tempo Discretizada														
Opções	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
g=1				1	1	1	1	1	1	1					
g=2					1	1	1	1	1	1	1				
g=3						1	1	1	1	1	1	1			
g=4							1	1	1	1	1	1	1		
g=5								1	1	1	1	1	1	1	
g=6									1	1	1	1	1	1	1

3.4.3 *Modelo de Programação de Frota em Escala de Tempo Discretizada, com Deslocamento*

O terceiro modelo, designado de “Modelo de Programação de Frota em Escala de Tempo Discretizada, com Deslocamento”, é uma extensão do modelo “Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão II”, contemplando os tempos de deslocamento das embarcações.

Para incluir os tempos de viagem introduziu-se a variável binária de fluxo z_{ij}^r , que será igual a 1 se a embarcação r se deslocar do local de término da tarefa i para o local de início da tarefa j . Toda vez que uma tarefa termina, impõe-se uma restrição que força a embarcação r a seguir para um próximo local, podendo este ser o local representativo de sua condição final.

Para que seja possível incorporar os tempos de deslocamento das embarcações, é necessário ajustar o parâmetro $tempo(i,j,r)$ para a escala de tempo escolhida. Este ajuste consiste em dividir o tempo do deslocamento de uma embarcação entre dois locais, cuja unidade é [dia], para a unidade de tempo adotada. Por exemplo, se o fator de discretização for igual a 3, a unidade de tempo passa a ser 8 horas. Neste caso, o tempo de viagem seria dividido por $\frac{1}{3}$ e, em seguida, arredondado para o inteiro mais próximo. Este processo poderá introduzir imprecisões nos tempos de viagem, que influenciarão os resultados gerados.

Assim como nos modelos de fluxo em rede, foi necessário definir uma variável que registra o instante de chegada de cada embarcação r no local de início da tarefa j , vindo da tarefa i : ch_{ij}^r .

Por último, cabe ressaltar que a notação de conjuntos de nós e arcos introduzida na seção 3.3 é aproveitada, sendo compatível com a formulação deste modelo.

O modelo passa a ser definido nos seguintes termos:

Variáveis de Decisão

$$w_j^g = \begin{cases} 1, & \text{se a tarefa } j \in N_2 \text{ iniciar no instante } inst(g), g \in conj_ini(j) \\ 0, & \text{em caso contrário} \end{cases}$$

$$x_{rj} = \begin{cases} 1, & \text{se a tarefa } j \in N_2 \text{ utilizar a embarcação } r \in E \\ 0, & \text{em caso contrário} \end{cases}$$

$$z_{ij}^r = \begin{cases} 1, & \text{se a embarcação } r \in E \text{ se desloca ao local de início da} \\ & \text{tarefa } j \in N_2 \cup N_3(r) \text{ após ter concluído a tarefa } i \in N_1(r) \cup N_2 \\ 0, & \text{em caso contrário} \end{cases}$$

Variáveis Auxiliares

$Atraso_j$ - tempo de atraso no início da execução da tarefa $j \in N_2$

$$y_{rj}^t = \begin{cases} 1, & \text{se a embarcação } r \in E \text{ estiver alocada à tarefa } j \in N_2 \\ & \text{no instante } t, 1 \leq t \leq T_{MAX} \\ 0, & \text{caso contrário} \end{cases}$$

ch_{ij}^r - instante de chegada da embarcação $r \in E$ ao local de início da tarefa $j \in N_2 \cup N_3(r)$ após ter concluído a execução da tarefa $i \in N_1(r) \cup N_2$

ini_j - instante de início efetivo da tarefa $j \in N_2$

Função Objetivo

$$\min CustoTotal = \sum_{j \in N} Atraso_j penal(j) + \sum_{r \in E} \sum_{i \in N_1(r) \cup N_2} \sum_{\substack{j \in N_2 \cup N_3(r) \\ j \neq i}} z_{ij}^r cv(r) tempo(i, j, r) \quad (3.85)$$

Restrições

$$\sum_{g \in \text{conj_ini}(j)} w_j^g = 1 \quad j \in N_2 \quad (3.86)$$

$$\sum_{g \in \text{conj_ini}(j)} \text{inst}(g)w_j^g \geq \sum_{h \in \text{conj_ini}(i)} \text{inst}(h)w_i^h + p(i) \quad i \in \text{pred}(j), j \in N_2 \quad (3.87)$$

$$\sum_{r \in \text{emb}(c)} x_{rj} \geq q(j, c) \quad j \in N_2, c \in C \wedge q(j, c) > 0 \quad (3.88)$$

$$\sum_{r \in E} x_{rj} \text{compat}(j, r) = \sum_{c \in C} q(j, c) \quad j \in N_2 \quad (3.89)$$

$$a(t, r)y_{rj}^t \geq \left(\sum_{g \in \text{conj_ini}(j)} m(g, t)w_j^g \right) + x_{rj} - 1 \quad j \in N_2, r \in E, t: li(j), \dots, ls(j) + p(j) - 1 \quad (3.90)$$

$$\sum_{j \in N} y_{rj}^t \leq 1 \quad r \in E, t: 1, \dots, T_{MAX} \quad (3.91)$$

$$\text{ini}_j = \sum_{g \in \text{conj_ini}(j)} \text{inst}(g)w_j^g \quad j \in N_2 \quad (3.92)$$

$$\text{Atraso}_j \geq \text{ini}_j - \text{inicio}(j) \quad j \in N_2 \quad (3.93)$$

$$\sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} z_{ij}^r = x_{rj} \quad r \in E, j \in N_2 \quad (3.94)$$

$$\sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} z_{ij}^r = \sum_{\substack{k \in N_2 \cup N_3(r) \\ k \neq j}} z_{jk}^r \quad r \in E, j \in N_2 \quad (3.95)$$

$$ch_{ij}^r \geq \text{disp}(r) + \text{tempo}(i, j, r) - (1 - z_{ij}^r)T_{MAX} \quad r \in E, i \in N_1(r), j \in N_2 \cup N_3(r) \quad (3.96)$$

$$ch_{ij}^r \geq \sum_{g \in \text{conj_ini}(i)} \text{inst}(g)w_i^g + p(i) + \text{tempo}(i, j, r) - (1 - z_{ij}^r)T_{MAX} \quad r \in E, i \in N_2, j \in N_2 \cup N_3(r), i \neq j \quad (3.97)$$

$$\sum_{g \in \text{conj_ini}(j)} \text{inst}(g)w_j^g \geq ch_{ij}^r \quad r \in E, i \in N_1(r) \cup N_2, j \in N_2, i \neq j \quad (3.98)$$

$$\text{Atraso}_j \geq 0, \text{ini}_j \geq 0 \quad j \in N_2 \quad (3.99)$$

$$x_{rj} \in \{0, 1\} \quad j \in N_2, r \in E \quad (3.100)$$

$$w_j^g \in \{0, 1\} \quad j \in N_2, 1 \leq g \leq \psi \quad (3.101)$$

$$y_{rj}^t \geq 0 \quad j \in N_2, r \in E, t: 1, \dots, T_{MAX} \quad (3.102)$$

$$ch_{ij}^r \geq 0, z_{ij}^r \in \{0, 1\} \quad r \in E, i \in N_1(r) \cup N_2, j \in N_2 \cup N_3(r), i \neq j \quad (3.103)$$

A eq. (3.85) é a função objetivo do problema, contendo as parcelas de penalidade por atraso e o custo variável. A restrição (3.86) impõe que um período de início, dentro da faixa que constitui a janela de tempo, deverá ser necessariamente escolhido para cada tarefa j . A restrição (3.87) faz com que o início de qualquer tarefa j apenas ocorra quando as tarefas predecessoras tiverem sido concluídas. A restrição (3.88) garante que a demanda por classe de embarcação seja atendida. Como é possível haver a cessão de embarcações entre classes, a restrição (3.89) impede que uma embarcação seja computada mais de uma vez, garantindo o atendimento da demanda quanto à totalidade de embarcações requeridas. A restrição (3.90) faz com que a variável y_{rj}^t assumo o valor 1 se a embarcação r iniciou a tarefa j ($x_{rj}=1$) em qualquer $inst(g)$ tal que $[m_{g,t}]=1$. Contudo, se a embarcação r não estiver afretada (disponível) em algum instante t ($[a_{t,r}]=0$), então necessariamente a embarcação não poderá ser designada a esta tarefa ($x_{rj}=0$). A restrição (3.91) impede que uma embarcação esteja alocada a mais de uma tarefa ao mesmo tempo. A restrição (3.92) define o instante efetivo de início da tarefa j . A restrição (3.93) calcula o atraso para início da tarefa, em relação ao instante desejado, dado pelo parâmetro $início(j)$. A restrição (3.94) garante que para cada embarcação r designada à tarefa j (por meio da variável x_{rj}), haja um fluxo de chegada z_{ij}^r vindo do nó de origem da embarcação r ou de alguma outra tarefa. A restrição (3.95) impõe a continuidade do fluxo das embarcações. A restrição (3.96) calcula os instantes de chegada das embarcações a uma tarefa, vindo da condição inicial. A restrição (3.97) calcula os instantes de chegada das embarcações a uma tarefa, vindo de outra tarefa, já concluída. A restrição (3.98) faz com que o início da tarefa seja maior ou igual aos instantes de chegada das embarcações. Por último, as restrições (3.99) e (3.103) impõem as condições de não-negatividade e integralidade nas variáveis do modelo.

Modelos similares ao modelo de programação de frota em escala de tempo discretizada, com aplicação em problemas de roteirização e programação de veículos com janela de tempo não foram encontrados na literatura. As aplicações mais próxima encontradas são alguns modelos de “job shop” (Morton; Pentico, 1993) em que a representação se dá por meio da discretização da escala de tempo²⁸. Outras representações clássicas de modelos “job shop”, em que a decisão da ordem de atendimento das tarefas (isto é, a definição da seqüência em cada máquina) está associada à definição do instante de tempo em que isto deverá ocorrer, são encontradas em Manne (1960), reapresentadas em Ravetti et al. (2007). Estas aplicações, contudo, não consideram alocação simultânea de recursos. Outra semelhança encontrada na literatura está no trabalho de Pritsker et al. (1969). Estes autores estudaram o problema de programação de múltiplos projetos (“multiproject scheduling”) com recursos limitados, propondo uma formulação matemática em um escala de tempo discretizada, com alocação simultânea de recursos e até mesmo consideraram a substituição de um recurso por outro equivalente. Não incorporaram, contudo, o tempo de “set up”. Outros trabalhos publicados na área de programação de projetos (“project scheduling”) têm alguma semelhança com o tema desta pesquisa (Brucker, et al.; 1999). Contudo, o objetivo destes problemas é a minimização do horizonte de planejamento (“makespan”). Estas aplicações, em geral, desconsideram os tempos de “set-up” – que, no caso desta pesquisa, equivaleria aos tempos de deslocamento das embarcações entre o local de término de uma tarefa e local de início da tarefa subsequente.

²⁸ Ver tópico “Exact Methods” do capítulo 15 “Scheduling Job Shops: Basic Methods” ; os autores também apresentam a formulação clássica usando restrições disjuntivas.

3.4.4 Extensões do Modelo

As mesmas extensões desenvolvidas para os modelos de fluxo em rede serão apresentadas para os modelos de tempo discreto.

3.4.4.1 Decisões de Afretamento

Para incluir as decisões de afretamento, é necessário utilizar o parâmetro, introduzido na seção 3.3, que indica quais embarcações do conjunto E que estão ativas:

$ativa(r)$ será igual a 1, se a embarcação r pertencer à frota e 0, em caso contrário (a embarcação é candidata).

Além disso, é necessário informar o custo de afretamento para cada embarcação:

$afret(r)$ será zero para as embarcações r que pertencem à frota e, para as embarcações candidatas, igual ao produto da taxa diária de afretamento pelo período de afretamento.

Uma nova variável de decisão é introduzida para escolher quais embarcações irão compor a frota:

$$v_r = \begin{cases} 1, & \text{se a embarcação } r \in E \text{ for utilizada} \\ 0, & \text{em caso contrário} \end{cases}$$

Cabe observar que sendo r uma embarcação já afretada, isto é, que já faz parte da frota, automaticamente v_r será igual a 1.

Dessa maneira, a função objetivo fica alterada para (3.104a) para os modelos em escala de tempo discretizada sem deslocamento e (3.104b) para o modelo em escala de tempo discretizada com deslocamento:

$$\min CustoTotal = \sum_{j \in N} Atraso_j penal(j) + \sum_{r \in E} v_r afret(r) \quad (3.104a)$$

$$\min CustoTotal = \sum_{j \in N} Atraso_j penal(j) + \sum_{r \in E} \sum_{i \in N_1(r) \cup N_2} \sum_{\substack{j \in N_2 \cup N_3(r) \\ j \neq i}} z_{ij}^r cv(r) tempo(i, j, r) + \sum_{r \in E} v_r afret_r \quad (3.104b)$$

As restrições necessárias para permitir a seleção e incorporação das embarcações na frota, válida para ambos os modelos, são as (3.105), (3.106) e (3.107), abaixo apresentadas.

$$x_{rj} \leq v_r \quad r \in E, j \in N \quad (3.105)$$

$$v_r \geq ativa(r) \quad r \in E \quad (3.106)$$

$$v_r \in \{0, 1\} \quad r \in E \quad (3.107)$$

A restrição (3.105) garante que, se a embarcação r não estiver ativa, ela não poderá ser alocada a nenhuma tarefa. A restrição (3.106) garante que as embarcações que já estão ativas, isto é, que pertencem à frota, sejam automaticamente habilitadas. Por último, a restrição (3.107) impõe que as variáveis sejam binárias.

3.4.4.2 Abastecimento e Troca de Tripulação

A inclusão das tarefas de abastecimento e troca de tripulação é simples, sendo uma tarefa igual às demais. Para isso, é necessário informar qual a embarcação deverá retornar à base e a correspondente janela de tempo em que isto deverá ocorrer. Para evitar que seja designada uma embarcação diferente à tarefa de abastecimento, é necessário atualizar o parâmetro de compatibilidade – $compat(j, r)$,

de forma que, apenas a embarcação r possa realizar a respectiva tarefa de abastecimento j . Portanto, as tarefas específicas de abastecimento terão o parâmetro $compat(j,r)=1$ para cada embarcação r designada a abastecer, e zero para as demais embarcações.

3.4.4.3 Agrupamento em Conjunto de Embarcações

A implementação do processo de seleção das embarcações nos modelos matemáticos, por meio do agrupamento das embarcações em conjuntos, requer que dois parâmetros adicionais sejam definidos:

$Conj = [conj_{h,r}]$ uma matriz com dimensão $\phi \times \varepsilon$, em que ϕ é o número de todos os possíveis conjuntos para as η tarefas do problema; ε é a quantidade de embarcações. Se o conjunto h contiver a embarcação r , então o elemento $[conj_{h,r}]$ será igual a 1 e 0, em caso contrário;

$conj_tar(j)$ contém os índices dos conjuntos de embarcações candidatas para realizar a tarefa j .

Além desses parâmetros, será empregada uma nova variável de decisão s_h que decidirá pela escolha do conjunto h . As restrições (3.61), (3.75) e (3.88) dos modelos em escala de tempo discretizada sem deslocamento – versão I, modelo em escala de tempo discretizada sem deslocamento – versão II e modelo em escala de tempo discretizada com deslocamento, respectivamente, podem ser eliminadas. Devem ser acrescentadas, a qualquer um dos três modelos, as restrições (3.108) a (3.110).

$$\sum_{h \in conj_tar(j)} s_h = 1 \quad j \in N \quad (3.108)$$

$$x_{rj} \leq \sum_{h \in conj_tar(j)} s_h * Conj(h,r) \quad r \in E, j \in N \quad (3.109)$$

$$s_h \in \{0,1\} \qquad 1 \leq h \leq \phi \qquad (3.110)$$

A restrição (3.108) garante que necessariamente um conjunto será escolhido para cada tarefa j . A restrição (3.109) limita o fluxo às embarcações que pertençam ao conjunto escolhido. A restrição (3.110) força que as variáveis sejam binárias.

3.5 Conclusão

Este capítulo teve for finalidade elaborar modelos matemáticos representativos do problema desta pesquisa. As representações desenvolvidas basearam-se nas estratégias clássicas de modelagem de problemas de fluxo em rede e em problemas de “job shop scheduling”. A complexidade do problema foi incorporada pelos modelos, sendo evidenciada pelo elevado número de variáveis binárias de decisão. Para que estes modelos possam gerar respostas satisfatórias, serão empregados alguns métodos de solução, apresentados no próximo capítulo.

4 MÉTODOS DE SOLUÇÃO

Este capítulo tem por objetivo apresentar métodos de solução desenvolvidos para o problema de PROGRAMAÇÃO DE FROTA DE APOIO A OPERAÇÕES “OFFSHORE” SUJEITA À REQUISIÇÃO DE MÚLTIPLAS EMBARCAÇÕES PARA UMA MESMA TAREFA.

4.1 Considerações Preliminares

Os modelos apresentados no capítulo 3 constituem representações matemáticas do problema que é objeto desta pesquisa. Quaisquer métodos de solução que forem desenvolvidos necessariamente terão que gerar respostas para as decisões presentes nestes modelos, segundo os objetivos fixados, e respeitando as restrições identificadas.

A revisão bibliográfica mostrou que duas abordagens distintas podem ser empregadas para resolução de problemas de programação de veículos, a saber: as estratégias otimizantes, ou exatas, e as estratégias heurísticas. Estas podem ser construtivas, de busca local e de busca estendida. A limitação computacional de resolver problemas de programação de veículos com janela de tempo utilizando métodos exatos advém do fato de tais problemas serem essencialmente combinatórios. Além disso, a incorporação de restrições de janela de tempo na formulação matemática reduz a qualidade dos limitantes gerados na versão linear (relaxada) deste modelo (Cordeau et al., 2005).

Apesar disso, será testado o algoritmo “branch and cut” que acompanha o pacote computacional de otimização CPLEX 10.0. Nos casos em que for possível a

obtenção de pelo menos uma solução inteira viável, poder-se-á estimar o valor do “gap”, isto é, a distância da solução obtida em relação ao melhor limitante inferior obtido por esse software. Este valor servirá de referência para avaliação do desempenho dos demais métodos de solução.

Com o objetivo de obter soluções iniciais viáveis para o problema desta pesquisa, será implementado e testado o método “feasibility pump” de Fischetti et al. (2005). Esta estratégia tem se mostrado promissora, por gerar soluções viáveis mediante a resolução de sucessivos problemas de programação linear, devidamente definidos.

Além do “pump”, será proposta uma heurística²⁹ construtiva probabilística, que define, em cada iteração, a “próxima” tarefa que será atendida. A escolha da tarefa dependerá da aplicação de uma regra de decisão sorteada dentre um conjunto de regras. Este sorteio levará em consideração o grau de criticidade das tarefas aguardando alocação. Por exemplo, tarefas em fila com o risco de violarem suas janelas de tempo terão prioridade em relação às outras tarefas; caso nenhuma tarefa corra o risco de tornar a solução inviável por violação de janela de tempo, a regra poderá ser outra. Duas versões serão apresentadas, a “básica” e a “gulosa”, diferindo apenas no critério de escolha das embarcações.

O método “branch and cut” será aplicado aos modelos de fluxo em rede, versões completa e compacta, e aos modelos de tempo discreto. A heurística, por sua vez, utilizará a representação do “modelo de fluxo em rede – conjuntos”, pela facilidade de gerenciar os conjuntos de embarcações em relação às embarcações individualmente, quando da implementação computacional. Cabe lembrar que as

²⁹ Heurística - arte de inventar, de fazer descobertas; ciência que tem por objeto a descoberta dos fatos – Dicionário eletrônico Houaiss

soluções heurísticas, quando houver, poderão ser refinadas com a aplicação subsequente de um método de melhoria.

Os métodos de busca em vizinhança comumente empregados em problemas de roteirização e programação de veículos, revisados por Bräysy; Gendreau (2005a), são baseados na geração da vizinhança de uma solução mediante operadores ou movimentos previamente definidos. Contudo, a estratégia proposta por Fischetti; Lodi (2003) consiste em explorar a vizinhança utilizando a estrutura matemática do problema. Esta vizinhança é definida por meio de novas restrições incorporadas ao modelo matemático que delimita a região de busca. Após a definição de um subespaço de solução do problema, esta região é explorada localmente por meio do algoritmo “branch and cut”, presente no pacote de otimização CPLEX 10.0. Este algoritmo informará se, para o tempo despendido na busca, foi encontrada alguma solução e se esta é ótima naquele subespaço. Em função da resposta obtida, o algoritmo “local branching” redefinirá o espaço de solução que, continuará sendo, em seguida, explorado.

Além da técnica “local branching”, será testada a heurística “variable neighborhood search” proposta por Mladenović; Hansen (1997), também baseada na exploração do espaço de solução definido pela estrutura matemática do problema, com a adição de novas restrições.

Todos estes métodos de solução serão testados para um conjunto de problemas descritos em detalhe no próximo capítulo.

4.2 Heurística Construtiva (H&C)

Esta heurística foi estruturada explorando as propriedades do problema em questão, programando, em cada iteração, uma tarefa da fila. Este algoritmo buscou

traduzir um processo tentativo, ou mesmo intuitivo, de priorizar as tarefas e alocar um conjunto de embarcações a cada uma delas, levando em conta a criticidade no atendimento da janela de tempo. Isto significa que, à medida que tarefas vão sendo programadas, passam a ter prioridade aquelas cujas janelas de tempo estão se esgotando. Todas as decisões tomadas se dão por meio de sorteios, escolhendo, inicialmente, a regra de decisão e, em seguida, a tarefa e o conjunto de embarcações.

A heurística construtiva foi desenvolvida visando, inicialmente, a geração de pelo menos uma solução viável para o problema de programação de frota. Assim, nas primeiras versões da heurística, o processamento era encerrado assim que uma solução viável fosse gerada.

Posteriormente, a heurística passou a ter como critério de parada o tempo máximo de processamento ou, se o “gap” da melhor solução gerada atingisse um patamar inferior a 1% (definido arbitrariamente). Para que isso seja possível, a heurística deve receber como parâmetro de entrada um limitante inferior da solução. Este limitante pode ser, por exemplo, o valor fornecido pelo pacote computacional CPLEX. O cálculo do “gap”, por parte da heurística construtiva, e a eventual interrupção de sua execução visa apenas a redução do tempo de processamento computacional.

Em virtude de sua natureza probabilística, a heurística é executada diversas vezes, quando são alterados, de maneira sistematizada, os pesos dos priorizadores (para sorteio das regras) e a semente do gerador de números aleatórios. A solução heurística é a melhor solução gerada dentro de um limite de tempo estipulado previamente.

A figura 4.1 ilustra, de maneira esquemática, o processo de construção da solução. Para gerenciar este processo, foram utilizados os parâmetros e variáveis

indicados na tabela 4.1. Os parâmetros e as variáveis específicos das regras de decisão para escolha das tarefas e das embarcações, serão posteriormente apresentados.

Tabela 4.1 – Lista de parâmetros e variáveis utilizados no loop principal da heurística construtiva

Parâmetro / Variável	Descrição
gap	gap da melhor solução gerada;
f_opt	valor da função objetivo da melhor solução gerada;
f_atual	valor da função objetivo da solução que está sendo construída;
x_opt	melhor solução gerada;
x_atual	solução que está sendo construída;
bound	limitante inferior da solução;
tempo_total_processamento	duração da execução do método heurístico;
k	número da iteração do algoritmo;
L	número de tarefas de um problema;

```

função Heurística Construtiva (tempo_total_processamento; bound)
0 início
1   gap:= +∞; f_opt:= +∞; x_opt:= indefinido;
2   enquanto (tempo < tempo_total_processamento) e (gap > 1%) fazer
3     k:=1; f_atual:= +∞; x_atual := indefinido;
4     enquanto (k ≤ L) fazer
5       escolher a k-ésima tarefa da fila segundo o critério de criticidade escolhido;
6       escolher o conjunto de embarcações para a k-ésima tarefa;
7       se (início da k-ésima tarefa respeitar a janela de tempo) então
8         atualizar x_atual, f_atual;
9       senão
10        x_atual:=indefinido; f_atual:= +∞; k:= L;
11      fim se;
12      k:=k+1;
13    fim_enquanto;
14    se (f_atual < f_opt) então
15      x_opt:=x_atual; f_opt:=f_atual;
16      atualizar gap;
17    fim_se;
18  fim_enquanto;
19 fim;

```

Figura 4.1 – Estrutura geral da heurística construtiva

A linha 1 inicializa o “gap” e as variáveis representativas da melhor solução do problema. Enquanto o tempo de processamento não atingir o tempo total de execução estabelecido e o “gap” for superior a 1%, o algoritmo executará o loop entre

as linhas 2 e 18. O loop entre as linhas 4 e 13 visa construir uma solução. Para isso, em cada passo do loop, uma tarefa da fila (isto é, uma tarefa ainda não programada) é escolhida, segundo um determinado critério (linha 5), assim como um conjunto de embarcações (linha 6). A eventual violação da janela de tempo advinda destas escolhas é avaliada na linha 7. Caso a viabilidade da solução seja mantida, as variáveis que armazenam a solução atual são atualizadas; caso contrário, este processo é interrompido. Em seguida, o valor da função objetivo da solução atual é avaliado, entre as linhas 14 e 17 e, caso seja inferior à melhor solução, esta é atualizada.

Visando formalizar a estruturação desta heurística, no que diz respeito ao detalhamento do processo decisório presente entre as linhas 5 a 12, empregou-se a notação apresentada na tabela 4.2.

Tabela 4.2 - Lista de parâmetros e variáveis utilizados no processo de decisão da heurística construtiva

Parâmetro / Variável	Descrição
N	conjunto de tarefas $\{1, \dots, n\}$;
i, j	índices do conjunto N ;
L	tamanho do conjunto N ($L= N $);
$OrdSeq(\text{Critério})$	conjunto N ordenado pelo critério informado;
$LI(i)$	limite inferior da janela de tempo da tarefa $i \in N$;
$LS(i)$	limite superior da janela de tempo da tarefa $i \in N$;
$InicioDesejado(i)$	início desejado da tarefa $i \in N$;
$p(i)$	duração da tarefa $i \in N$;
$dist(i,j)$	distância (milhas) entre o local de término da tarefa i e o local de início da tarefa j ;
E	conjunto de embarcações $\{1, \dots, \varepsilon\}$;
R	índice do conjunto E ;
$v(r)$	velocidade da embarcação r , em nós;
$CV(r)$	custo variável da embarcação r ;
$PosEmb(r)$	posição da embarcação r (no início da simulação, indica o local em que a embarcação encontra-se disponível);
$EmbLivre(r)$	instante em que a embarcação r ficará livre (no início da simulação, indica o instante de disponibilidade inicial da embarcação);

Tabela 4.2 - Lista de parâmetros e variáveis utilizados no processo de decisão da heurística construtiva – Continuação

Parâmetro / Variável	Descrição
$Conj$	lista contendo todos os subconjuntos de embarcações de todas as tarefas;
H	índice de $Conj$;
$c(i)$	número de conjuntos de embarcações da tarefa $i \in N$;
Tol	conjunto de tarefas com tolerância negativa;
$nTol$	número de tarefas com tolerância negativa ($nTol = Tol $);
$Cheg$	conjunto de tarefas disponíveis para serem programadas;
$nCheg$	número de tarefas disponíveis para serem programadas ($nCheg = Cheg $);
Pav	duração médias das tarefas ainda não programadas;
$p1$	probabilidade de escolha da regra de tolerância negativa;
$p2$	probabilidade de escolha da regra de menor folga;
$p3$	probabilidade de aplicação da regra gulosa na escolha do conjunto de embarcações;
$lim1$	número máximo de tarefas que participam do sorteio para retirada da fila, dentre um conjunto de tarefas ordenado por ordem de chegada;
$lim2$	coeficiente usado na determinação do fator de <i>segurança</i> ;
$lim3$	número máximo de tarefas que participam do sorteio para retirada da fila, dentre um conjunto de tarefas ordenado pelo critério de menor folga ou menor priorizador;
$lim4$	número de refinamentos na escolha do conjunto de embarcações;
<i>segurança</i>	coeficiente de segurança, igual a $(1+lim2)$;
$atraso(i)$	atraso da tarefa $i \in N$;
$Folga(i)$	folga da tarefa $i \in N$;
$Tolerância(i)$	tolerância da tarefa $i \in N$;
$Priorizador(i)$	medida de “urgência-disponibilidade” da tarefa $i \in N$;
$PossivelInicio(i)$	possível início da tarefa $i \in N$;
$InicioConjunto(h)$	possível início de uma tarefa pelo conjunto h ;
$nConjViaveis(i)$	número de conjuntos viáveis da tarefa $i \in N$;
$ConjCedo(i)$	índices dos conjuntos de embarcações que iniciam a tarefa $i \in N$ no instante mais cedo;
$nConjCedo(i)$	número de conjuntos de embarcações que iniciam a tarefa $i \in N$ no instante mais cedo ($nConjCedo(i) = ConjCedo(i) $);
$Tarefa(h)$	índice da tarefa correspondente ao conjunto h ;

A seguir, será discutido em detalhe o processo de decisão para escolha da “próxima” tarefa que será programada, juntamente com a definição do conjunto de embarcações. Este detalhamento será dividido em três partes: a lista de parâmetros

que dá subsídio a estas escolhas, o processo de escolha do conjunto de embarcações, diferenciando as estratégias básica e gulosa, e o processo de escolha das tarefas.

4.2.1 *Parâmetros e variáveis da heurística*

Os parâmetros apresentados em seguida são os utilizados na definição da regra de decisão que a heurística empregará e auxiliarão o processo de sorteio de qual tarefa deverá ser retirada da fila e de qual conjunto de embarcações deverá ser alocado:

1. *Início-Conjunto*: uma determinada tarefa, quando encerrada, libera um conjunto de embarcações, fazendo com que cada tarefa em fila seja uma potencial candidata a ser programada, desde que as restrições de precedência sejam respeitadas. Uma forma de avaliar quais tarefas estariam aptas para iniciar é recalculando o possível instante de início de cada tarefa utilizando seus conjuntos de embarcações. Este cálculo consiste em atualizar o possível instante de início apenas para os conjuntos que contenham alguma embarcação recém-liberada (isto é, que estava sendo utilizada na tarefa que foi concluída). O instante de início dos conjuntos (de cada tarefa), representado pelo parâmetro *InicioConjunto(h)*, dependerá do instante em que cada embarcação do conjunto ficou liberada de sua tarefa anterior, e do tempo para se reposicionar até o local de início da tarefa em fila.
2. *Possível-Início*: após os conjuntos das tarefas em fila terem sido atualizados quanto ao possível instante de início, o parâmetro *PossivelInicio(i)* receberá o menor instante dentre todos os possíveis instantes de início dos conjuntos de embarcações da tarefa *i*. Este parâmetro permitirá avaliar o grau de criticidade de uma tarefa quanto à janela de tempo. Pode ocorrer, inclusive, que o instante mais cedo de início seja maior que o limite superior da janela de tempo, indicando que a solução é inviável.

3. *Folga*: após a definição do possível instante de início, cada tarefa i deverá ser avaliada quanto à folga ($Folga(i)$) existente para iniciar a operação, sendo ela definida como a diferença entre o limite superior da janela de tempo ($LS(i)$) e o possível instante de início ($PossivelInicio(i)$). Se ocorrer folga negativa para alguma tarefa em fila, então a alocação parcialmente gerada leva a uma solução inviável; em caso contrário, pode-se utilizar este parâmetro para priorizar as tarefas com menor folga ou, na comparação da folga de cada tarefa com a média das durações das tarefas em fila, que é a definição de “tolerância”.
4. pav : este parâmetro indica a duração média das tarefas em fila.
5. *Tolerância*: este parâmetro, calculado para cada tarefa em fila, é definido como sendo a folga da tarefa menos a duração média (pav) multiplicada por um fator de *segurança*, e mede o risco de haver violação da janela de tempo da tarefa quando outra tarefa da fila é programada antes dela. Uma tolerância negativa indica que a tarefa deve ser priorizada quanto à programação³⁰. O cálculo da tolerância é seguido pela atualização da variável $nTol$ que mede quantas tarefas possuem tolerância negativa.
6. *Urgência-Disponibilidade*: este parâmetro, definido para cada tarefa i da fila, denominado de “priorizador” ($Priorizador(i)$) é definido como sendo o produto da tolerância ($Tolerancia(i)$) pelo possível instante de início ($PossivelInicio(i)$). Esta medida ajuda resolver situações em que uma tarefa possui baixa tolerância, mas não pode iniciar por não ter as embarcações disponíveis. Uma tarefa com maior tolerância, mas cuja data possível de início é mais cedo (por ter as embarcações disponíveis) poderia iniciar ao invés de ficar retida, aguardando que a outra seja programada. Portanto, segundo este critério, quem tiver o menor valor é priorizado.

³⁰ Existem algumas limitações deste conceito: 1) quando há tarefas com durações muito díspares, a média pode não ser um bom estimador; 2) quando a frota é muito grande, haverá possibilidade de realização de mais de uma tarefa simultaneamente, e uma baixa tolerância não significa que há risco real de violação da janela de tempo.

7. *nCheg*: esta variável verifica quantas tarefas, dentre as que ainda não foram programadas, possuem o possível instante de início superior ao limite inferior de sua janela de tempo. Se isto ocorrer, significa que a tarefa já está disponível para ser programada. Caso nenhuma tarefa atenda a esta condição, tem-se uma situação em que interessa priorizar as tarefas de acordo com a ordem de chegada prevista, já que nenhuma estaria ainda disponível.

4.2.2 *Sorteio do conjunto de embarcações & diferenciação entre as versões básica e gulosa da heurística construtiva*

Após a escolha de uma tarefa da fila, com base nas regras que serão posteriormente apresentadas, duas estratégias alternativas são empregadas para escolher o conjunto de embarcações que irá executar a tarefa, a saber: a estratégia básica e a estratégia gulosa.

A estratégia básica consiste em sortear um conjunto de embarcações que seja viável, isto é, que seja capaz de iniciar a tarefa respeitando o limite superior da janela de tempo. Após esta escolha, é sorteado um número de refinamentos m , o qual obriga ocorrer tantos sorteios de conjuntos de embarcações quantos necessários, para que o instante de início da tarefa seja refinado m vezes. O conceito de refinamento é o de identificar (novos) conjuntos cujos instantes de início sejam menores ou iguais ao instante de início do conjunto anteriormente sorteado. Por exemplo, se na primeira tentativa de refinamento for sorteado um conjunto cujo o início seja inferior ao instante de início fornecido pelo conjunto inicial, então este novo valor será a referência para os próximos $m-1$ refinamentos. Neste processo, apenas o instante de início é comparado, podendo haver eventual repetição do conjunto. Esta forma de sorteio das embarcações faz com que haja maior possibilidade de diversificação das soluções, ao mesmo tempo que força a escolha de conjuntos com instantes de início

próximos ao instante mais cedo possível, como consequência do processo de refinamento.

A estratégia gulosa consiste em forçar a escolha do conjunto de embarcações capaz de iniciar a tarefa no instante mais cedo. Dado que uma penalidade incide sobre o atraso para início da tarefa, é razoável que seja escolhido, dentre os conjuntos viáveis, aquele que inicia a tarefa mais rapidamente. Caso haja empate (mais de um conjunto é capaz de iniciar a tarefa no instante mais cedo), a regra prevê o sorteio equiprovável entre os conjuntos. Na implementação da heurística construtiva – versão gulosa, a regra gulosa não é aplicada para todas as tarefas da fila, pois isso poderia torná-la uma regra rígida. Assim, a regra gulosa é aplicada com uma probabilidade conhecida; caso não seja aplicada, recorre-se à estratégia básica, sorteando-se um conjunto viável e, subsequentemente, aplicando o processo de refinamento.

4.2.3 Sorteio da tarefa

O procedimento de definição da sequência de atendimento, ou seja, da escolha das tarefas, é apresentado na figura 4.2.

O processo de construção de uma solução é iniciado sorteando-se uma dentre as $lim1$ primeiras tarefas em fila, ordenadas quanto à ordem de chegada. Caso a tarefa sorteada contenha restrição de precedência, este processo é repetido. O parâmetro $lim1$, ao ser aumentado para $lim1=|N|$, faz com que qualquer tarefa possa ser sorteada no início do processo de programação. O intuito de utilizar valores maiores para $lim1$ é o de diversificar as possíveis sequências de alocação ou atendimento da demanda.

Figura 4.2 – Fluxograma do procedimento de escolha das regras de decisão

Após a escolha desta tarefa, a heurística irá decidir pela aplicação da regra gulosa ou da regra básica para escolha do conjunto de embarcações. Os tempos de viagem das embarcações do conjunto deverão ser calculados para que se possa determinar o possível instante de início da tarefa e, verificar se este instante respeita os limites da janela de tempo. Caso as embarcações cheguem antes do limite inferior, estas deverão aguardar; caso cheguem após o limite superior, ter-se-á caracterizado uma inviabilidade e o algoritmo encerra. Sendo uma chegada viável, passa-se a atualizar todos os parâmetros e variáveis de controle do problema, como o

InicioConjunto(h), *PossivelInicio(i)*, *Folga(i)*, *pav*, *Tolerancia(i)*, *Priorizador(i)*, *Cheg*, anteriormente definidos.

Após estes cálculos, a escolha da próxima tarefa se dará avaliando, em primeiro lugar, se $nCheg > 0$, isto é, se há tarefas prontas para serem alocadas. O conjunto *Cheg* contém as tarefas que estão aguardando programação (diferentemente das tarefas que ainda não estão disponíveis, por terem janelas de tempo de atendimento muito superior ao instante de tempo em que está ocorrendo o programação). Se $nCheg = 0$, então o critério de escolha volta a ser o empregado quando do início da simulação: ordena-se as tarefas por ordem de chegada e sorteia-se uma dentre as *lim1* primeiras tarefas.

Caso contrário, há tarefas ainda não programadas em fila, e o algoritmo deverá aplicar um critério de retirada da fila. A primeira checagem é quanto ao número de tarefas que têm tolerância negativa, ou seja, que correm risco de violarem suas janelas de tempo caso outras tarefas sejam escolhidas primeiro. Caso haja tarefas nesta situação, sorteia-se uma das duas regras: a) escolher a tarefa com maior tolerância negativa; b) escolher uma tarefa dentre as tarefas com tolerância negativa.

O outro caso, é quando há tarefas em fila e nenhuma delas possui tolerância negativa. A decisão de retirada da fila será feita avaliando-se: a) tarefas com menor folga; b) tarefas com menor priorizador. Após o sorteio da estratégia a ser seguida, sorteia-se uma dentre as *lim3* tarefas ordenadas segundo o critério escolhido.

Após a escolha da tarefa, repete-se o procedimento de escolha do conjunto, até que todas as tarefas tenham sido programadas. A figura 4.3 lista o pseudocódigo da heurística construtiva.

Heurística Construtiva

```

1 Passo 1: // Primeira tarefa ou quando não há tarefa em fila
2 atualiza OrdSeq(Data Chegada);
3  $m := \text{disc}(1, \text{mín}(l, \text{lim1}))$ ;
4  $i := \text{OrdSeq}(m)$ ;
5 ir para Passo 3;
6 Passo 2: se ( $nTol > 0$ ) então
7      $p := \text{inv.unif}(0;1)$ ;
8     se ( $p < p1$ ) então
9         // Escolhe a tarefa com menor tolerância negativa
10         $i := \text{argmin}\{\text{Tolerância}(j)\} \quad j \in \text{Tol}$ ;
11    se não
12        // Sorteia dentre as tarefas com tolerância negativa
13        atualiza OrdSeq(Tolerância);
14         $m := \text{disc}(1, nTol)$ ;
15         $i := \text{OrdSeq}(m)$ ;
16    fim se;
17    ir para Passo 3;
18 se não
19     $p := \text{inv.unif}(0;1)$ ;
20    se ( $p < p2$ ) então
21        // Sorteia dentre as tarefas com menor folga
22        atualiza OrdSeq(Folga);
23         $m := \text{disc}(1, \text{mín}(l, \text{lim3}))$ ;
24         $i := \text{OrdSeq}(m)$ ;
25    se não
26        // Sorteia dentre as tarefas com menor priorizador
27        atualiza OrdSeq(Priorizador);
28         $m := \text{disc}(1, \text{mín}(l, \text{lim3}))$ ;
29         $i := \text{OrdSeq}(m)$ ;
30    fim se;
31    ir para Passo 3;
32 fim se;
33 Passo 3:  $p := \text{inv.unif}(0;1)$ ;
34 se ( $p < p3$ ) então
35    // Escolhe o melhor conjunto de embarcações (regra gulosa)
36    atualiza OrdSeq(InicioConjunto( $h \quad \forall h : \text{Tarefa}(h)=i$ ))
37     $m := \text{disc}(1, n\text{ConjCedo}(i))$ 
38     $h := \text{OrdSeq}(m)$ 
39 se não
40    // Escolhe um conjunto viável e determina o número de refinamentos
41     $m := \text{disc}(1, \text{lim4})$ ;
42     $h := \text{Refinamento}(i; m)$ ;
43 fim se;
44  $\text{Inicio}(i) \leftarrow \text{InicioConjunto}(h)$ ;
45 se ( $\text{Inicio}(i) < \text{LI}(i)$ ) então  $\text{Inicio}(i) := \text{LI}(i)$ ;
46 se ( $\text{Inicio}(i) > \text{LS}(i)$ ) então ir para Passo 5;
47 se ( $\text{Inicio}(i) > \text{InicioDesejado}(i)$ ) então  $\text{atraso}(i) := \text{Inicio}(i) - \text{InicioDesejado}(i)$ ;
48  $N \leftarrow N \setminus \{i\}$ ;
49  $l := l - 1$ ;
50 se ( $l = 0$ ) então ir para o Passo 4;
51 atualiza:
52      $\text{EmbLivre}(r) = \text{Inicio}(i) + p(i)$ ;  $\forall r \in \text{Conj}(h)$ 
53      $\text{PosEmb}(r) = i$ ;  $\forall r \in \text{Conj}(h)$ 
54      $\text{InicioConjunto}(h)$   $\forall h : \text{Tarefa}(h) \in N$  onde:
55      $\text{InicioConjunto}(h) = \text{argmax}\{ \text{EmbLivre}(r) +$ 
56          $\text{dist}(\text{PosEmb}(r), \text{LocalInicio}(\text{Tarefa}(h))) / 24 * v(r) \} \quad \forall r \in \text{Conj}(h)$ 
57      $\text{PossivelInicio}(i) := \text{argmin}\{\text{InicioConjunto}(h)\} \quad \forall h : \text{Tarefa}(h)=i; \forall i \in N$ 
58      $\text{ConjCedo}(i) = \{h : (\text{Tarefa}(h)=i) \cap (\text{InicioConjunto}(h) = \text{PossivelInicio}(i))\}$ ;

```

57	$nConjCedo(i) = ConjCedo(i) ;$	
58	$pav := media(p(i));$	$\forall i \in N$
59	$Folga(i) = LS(i) - PossivelInicio(i);$	$\forall i \in N$
60	$Tolerancia(i) = Folga(i) - pav * seguran\c{c}a;$	$\forall i \in N$
61	$Priorizador(i) = Folga(i) * PossivelInicio(i);$	$\forall i \in N$
62	$Tol = \{i \in N: Tolerancia(i) < 0\};$	
63	$nTol = Tol ;$	
64	$Cheg = \{i \in N: PossivelInicio(i) > LI(i)\};$	
65	$nCheg = Cheg ;$	
66	se $(nCheg > 0)$ ent\c{a}o ir para Passo 2;	
67	se n\c{a}o ir para Passo 1;	
68	Passo 4: Fim => Solu\c{c}\c{a}o Vi\c{a}vel;	
69	Passo 5: Fim => Solu\c{c}\c{a}o Invi\c{a}vel;	

Figura 4.3 – Pseudo c\c{od}igo da heur\c{ist}ica construtiva

4.3 “Feasibility Pump”

O m\c{e}todo “feasibility pump” (FP), descrito no cap\c{it}ulo 2, \c{e} uma heur\c{ist}ica construtiva que parte de uma solu\c{c}\c{a}o invi\c{a}vel de um MIP, e busca, em cada itera\c{c}\c{a}o, achar o ponto mais pr\c{o}ximo que perten\c{c}a ao poliedro convexo do problema de programa\c{c}\c{a}o linear relaxado do problema original. Ao encontrar um ponto deste poliedro, o mesmo poder\c{a} n\c{a}o ser vi\c{a}vel para o MIP. A verifica\c{c}\c{a}o se d\c{a} pelo arredondamento para o inteiro mais pr\c{o}ximo de todas as vari\c{a}veis inteiras. Quando a itera\c{c}\c{a}o subsequ\c{e}nte a um arredondamento indicar que a dist\c{a}ncia entre este ponto “externo” \c{a} regi\c{a}o vi\c{a}vel e o ponto pertencente ao poliedro for nula, ent\c{a}o uma solu\c{c}\c{a}o vi\c{a}vel foi obtida, dado que houve converg\c{e}ncia para um v\c{e}rtice vi\c{a}vel.

A implementa\c{c}\c{a}o utilizada nesta pesquisa \c{e} baseada na proposta de Achterberg; Berthold (2005)³¹, em que o processo de converg\c{e}ncia \c{e} influenciado pela fun\c{c}\c{a}o objetivo do problema original, de forma a produzir solu\c{c}\c{a}o es vi\c{a}veis e, ao mesmo tempo, de alta qualidade. Estes autores denominaram esta varia\c{c}\c{a}o como “objective feasibility pump”.

³¹ O c\c{od}igo fonte em C++ usado nesta pesquisa foi cedido por Fischetti e Lodi.

Os principais parâmetros e variáveis empregadas na implementação deste método estão listados na tabela 4.3; o pseudo-código é mostrado na figura 4.4.

Tabela 4.3 - Lista de parâmetros e variáveis utilizados no “feasibility pump”

Parâmetro / Variável	Descrição
x^*	solução do problema de programação linear;
$\tilde{x}$	solução inteira obtida pelo processo de arredondamento;
S	conjunto auxiliar; pode se referir tanto ao conjunto de variáveis binárias quanto ao conjunto de variáveis inteiras (não binárias) do MIP;
B	conjunto de variáveis binárias do MIP;
I	conjunto de variáveis inteiras, não binárias, do MIP;
t	contador da iteração;
$maxIter$	variável que recebe o número máximo de iterações de cada estágio;
$maxIter_{ST1}$	número máximo de iterações do Estágio 1;
$maxIter_{ST2}$	número máximo de iterações do Estágio 2;
$maxStalls$	variável que recebe o número máximo de iterações sem melhoria permitidas em cada estágio;
$maxStalls_{ST1}$	número máximo de iterações sem melhoria do Estágio 1;
$maxStalls_{ST2}$	número máximo de iterações sem melhoria do Estágio 2;
$reiniciar$	variável que armazena o total de perturbações realizadas em cada estágio;
L	vetor que armazena o histórico de soluções geradas, com os respectivos pesos, ao longo de cada estágio;
T	número de variáveis que deverão ser modificadas;
δ_α	tolerância para diferenciação entre os pesos α ;
φ	fator redutor do peso α ;
$maxReinicios$	número máximo de reinícios possíveis no Estágio 2;

A implementação proposta pelos autores consiste em dividir o processamento em três estágios. No primeiro estágio, o conjunto S recebe o conjunto B , isto é, o conjunto de variáveis binárias do problema. Todos os cálculos de distância e arredondamento serão feitos considerando apenas as variáveis binárias da formulação matemática; as variáveis inteiras não-binárias não contribuem para os cálculos mencionados. No segundo estágio, todas as variáveis inteiras são contempladas no cálculo da distância. No terceiro estágio, caso uma solução viável

ainda não tenha sido gerada, recorre-se ao método “branch and cut”, usando como solução de referência a de menor distância do poliedro convexo.

Método “Objective Feasibility Pump”

Estágio 1

1. Inicializar $x^* := \arg \min \{c^T x \mid x \in P\}$, $\tilde{x} := [x^*]^B$, $S := B$, $t := 0$, $maxIter := maxIter_{ST1}$, $maxStalls := maxStalls_{ST1}$, $reiniciar := 0$, $L = \emptyset$
2. Se $\tilde{x}$ não mudou na última iteração, arredondar as T variáveis x_j^* mais fracionárias para o valor oposto comparado com $\tilde{x}_j$
3. Se existir o par $(\tilde{x}', \alpha_{t'}) \in L$ com $\tilde{x}' = \tilde{x}$ e $\alpha_{t'} - \alpha_t \leq \delta_\alpha$, realizar uma perturbação aleatória em $\tilde{x}$; atualizar $reiniciar := reiniciar + 1$
4. Se $\tilde{x}$ for viável para o MIP, parar
5. Fazer $L := L \cup \{(\tilde{x}, \alpha_t)\}$; fazer $t := t + 1$ e $\alpha_t := \varphi \alpha_{t-1}$
6. Se $t > maxIter$, ir para o próximo estágio
7. Resolver $x^* := \arg \min \{\Delta_\alpha(x, \tilde{x}) \mid x \in P\}$
8. Se $x^* = \tilde{x}$, então ir para o próximo estágio
9. Se $f(x^*)$ não diminuiu por pelo menos 10% nas últimas $maxStalls$ iterações, ir para o próximo estágio
10. Fazer $\tilde{x} := [x^*]^S$; Ir para o Passo 2

Estágio 2

1. Inicializar $\tilde{x}$ como sendo a solução inteira do Estágio 1 com menor valor $\Delta(x, \tilde{x})$, $x \in P$. Fazer $t := 0$, $S := I$, $maxIter := maxIter_{ST2}$, $maxStalls := maxStalls_{ST2}$, $reiniciar := 0$, $L = \emptyset$
2. Realizar os passos 2 a 10 do Estágio 1; se $reiniciar > maxReinicios$, então ir para o Estágio 3

Estágio 3

1. Aplicar o método “branch and cut” para resolver o MIP, sendo $\tilde{x}$ a solução inteira do Estágio 2 com menor valor $\Delta(x, \tilde{x})$, $x \in P$, a referência para elaboração da função distância. Parar após encontrar a primeira solução viável.

Figura 4.4 – Pseudo código do algoritmo “Feasibility Pump” (adaptado de Achterberg; Berthold, 2005)

Na linha 1 (Estágio 1), as variáveis do problema são inicializadas. A solução de partida x^* será a solução do problema de programação linear. Na linha 2, é checado se ocorreu ciclagem, comparando a solução inteira corrente com a gerada na iteração anterior. Caso isso aconteça, são escolhidas as T variáveis mais fracionárias da solução x^* , as quais serão modificadas. Na linha 3, também é verificado se alguma solução recém gerada indica a formação de ciclagem; neste caso, o procedimento será

introduzir uma perturbação aleatória na solução inteira corrente $\tilde{x}$. Na linha 4, é checado se a solução inteira (arredondada) é viável para o problema original. Na linha 5, a solução inteira é armazenada juntamente com o peso α utilizado na função objetivo. Além disso, o peso α é reduzido geometricamente por meio do parâmetro φ ($0 < \varphi < 1$). Na linha 6, caso o número de iterações tenha excedido o limite previsto, o próximo estágio é iniciado. Em caso contrário, o problema de programação linear relaxado, com a função objetivo modificada – reescrita utilizando-se o novo α e a solução inteira arredondada (inviável) $\tilde{x}$ – é resolvido, na linha 7. Caso a solução obtida x^* seja igual à solução de referência $\tilde{x}$, o próximo estágio é iniciado. Como no Estágio 1 apenas as variáveis binárias são utilizadas para o cálculo da distância, a obtenção de uma solução de distância nula não necessariamente significa que a solução obtida é viável para o problema original, sendo necessário contemplar as demais variáveis inteiras, se houver. Caso o valor da função objetivo da solução linear x^* indicar que não houve progresso na redução da distância de, pelo menos 10% nas últimas *maxStalls* iterações, o próximo estágio é iniciado, na linha 9. Em caso contrário, na linha 10, a solução x^* é arredondada, e o algoritmo retorna para o passo descrito na linha 2.

O Estágio 2 consiste em aplicar o Estágio 1, incluindo, na função distância, todas as variáveis inteiras. Um novo critério de parada é introduzido, que é o máximo número de reinícios permitidos, isto é, o máximo número de perturbações realizadas em caso de ciclagem – esta condição será checada na linha 3 do Estágio 1.

Caso nenhuma solução viável para o MIP original seja obtida após o Estágios 1 e 2, o Estágio 3 consistirá em recorrer ao algoritmo “branch and cut”, tendo como referência a solução de menor distância ao poliedro convexo.

4.4 “Local Branching” (LB)

A técnica “local branching” (LB), descrita no capítulo 2, é um método de busca que explora subespaços de solução do problema, definidos pela adição de novas restrições matemáticas. Após a definição desta região, um algoritmo de enumeração implícita como o “branch and bound” ou “branch and cut” é aplicado, com o objetivo de encontrar a melhor solução contida naquela região.

Como a forma de exploração da vizinhança se dá pela aplicação de um algoritmo de enumeração implícita, a implementação do “local branching” irá necessariamente interagir com um software de otimização matemática. O pseudocódigo do algoritmo LB, que gerencia a adição e remoção de novas restrições matemáticas, interagindo com o pacote de otimização CPLEX 10.0, é apresentado na figura 4.5³². Os principais parâmetros e variáveis utilizados (maiores detalhes em Fischetti; Lodi, 2003) são apresentados na tabela 4.4.

Tabela 4.4 - Lista de parâmetros e variáveis utilizados no “local branching”

Parâmetro / Variável	Descrição
<i>diversifica</i>	indica se o LB deve empregar a estratégia de diversificação;
<i>dv_max</i>	número máximo de diversificações possíveis;
<i>k</i>	parâmetro que define tamanho da vizinhança a ser explorada;
<i>limite_tempo_nó</i>	máximo tempo de processamento de um sub-problema;
<i>LT</i>	limite de tempo de processamento de um problema;
<i>melhorUB</i>	melhor limitante superior (“upper bound”) obtido;
<i>opt</i>	variável que indica se a solução final é ótima;
<i>primeira_solução</i>	indica se o algoritmo “branch and cut” deve ou não parar após encontrar a primeira solução;
<i>rhs</i>	parâmetro que expressa o tamanho da vizinhança na iteração;
<i>stat</i>	status da solução (ótima, inviável, viável, não encontrou);
<i>tempo_total_processamento</i>	máximo tempo de processamento do algoritmo LB;
<i>UB</i>	limitante superior de um problema;
<i>x*</i>	melhor solução de um problema;
$\tilde{x}$	solução corrente de um problema;

³² O código fonte em C usado nesta pesquisa foi cedido por Fischetti e Lodi.


```

função LocalBranching(k, tempo_total_processamento, limite_tempo_nó, dv_max, x*)
0 início
1   rhs := melhorUB := UB := LT := +∞; x* := indefinido;
2   opt := primeira_solução := verdadeiro; dv:=0; diversifica := falso;
3   repetir
4     se (rhs<∞) então adicionar restrição de ramificação local  $\Delta(x, \bar{x}) \leq rhs$  fim_se;
5     LT := mín (LT, tempo_total_processamento - tempo_gasto);
6     stat := MIPSOLVE(LT, UB, primeira_solução,  $\tilde{x}$ )
7     LT := limite_tempo_nó;
8     se (stat = "encontrou solução ótima") então
9       se ( $c^T \tilde{x} < melhorUB$ ) então  $melhorUB := c^T \tilde{x}$ ;  $x^* = \tilde{x}$  fim_se;
10      se (rhs≥∞) finalizar(opt);
11      reverter última restrição de ramificação local para  $\Delta(x, \bar{x}) \geq rhs + 1$ ;
12      diversifica := primeira_solução := falso;  $\bar{x} = \tilde{x}$ ;  $UB := c^T \tilde{x}$ ; rhs:= k;
13    fim_se
14    se (stat = "comprovada a inexistência de solução viável") então
15      se (rhs≥∞) finalizar(opt);
16      reverter última restrição de ramificação local para  $\Delta(x, \bar{x}) \geq rhs + 1$ ;
17      se (diversifica=verdadeiro) então
18        UB:=LT:= +∞; dv=dv+1; primeira_solução:=verdadeiro
19      fim_se;
20      rhs = rhs + < k/2 >; diversifica := verdadeiro;
21    fim_se
22    se (stat = "encontrou solução viável") então
23      se (rhs<∞) então
24        se (primeira_solução =verdadeiro) então
25          remover última restrição de ramificação local
26        senão
27          substituir última restrição de ramificação local por  $\Delta(x, \bar{x}) \geq 1$ 
28        fim_se
29      fim_se;
30      REFINAR( $\tilde{x}$ )
31      se ( $c^T \tilde{x} < melhorUB$ ) então  $melhorUB := c^T \tilde{x}$ ;  $x^* = \tilde{x}$  fim_se;
32      diversifica := primeira_solução := falso;  $\bar{x} = \tilde{x}$ ;  $UB := c^T \tilde{x}$ ; rhs:= k;
33    fim_se
34    se (stat = "não encontrou solução viável") então
35      se (diversifica =verdadeiro) então
36        substituir última restrição de ramificação local por  $\Delta(x, \bar{x}) \geq 1$ 
37        UB := LT := +∞; dv=dv+1; rhs = rhs + < k/2 >;
38        primeira_solução :=verdadeiro
39      senão
40        remover última restrição de ramificação local; rhs = rhs - < k/2 >
41      fim_se;
42      diversifica := verdadeiro;
43    fim_se;
44    até que (tempo_gasto> tempo_total_processamento) ou (dv>dv_max)
45    LT := tempo_total_processamento - tempo_gasto; primeira_solução := falso;
46    stat := MIPSOLVE(LT, melhorUB, primeira_solução, x*)
47    opt := (stat="encontrou solução ótima") ou
48          (stat = "comprovada a inexistência de solução viável")
49    finalizar(opt)
50 fim.

```

Figura 4.5 – Pseudo código do algoritmo “Local Branching” (adaptado de Fischetti;

Lodi, 2003)

O algoritmo é inicializado nas linhas 1 e 2. Da linha 3 à linha 44 ocorre o loop principal do LB. Exceto na primeira iteração, quando o algoritmo é inicializado, adiciona-se sempre, na linha 4, uma restrição de ramificação local, definindo a vizinhança que será explorada.

O ‘MIPSOLVE’ é o comando interno do CPLEX que chama a rotina de “branch and cut”, na linha 6, necessitando, para isso, quatro parâmetros de entrada: **LT** – o limite de tempo de processamento que, até que a primeira solução seja obtida, é igual ao tempo total de processamento³³; depois este parâmetro será igual ao tempo de processamento daquele nó ou o valor de tempo remanescente de processamento do LB, o que for menor; **UB** – o limitante superior da solução; **primeira_solução** – sendo verdadeiro, faz com que a exploração da árvore “branch and bound” ou “branch and cut” seja interrompida após a obtenção da primeira solução viável; caso seja falso, a exploração irá ocorrer até que LT seja atingido; $\tilde{x}$ – inicialmente indefinido, devolverá o valor da solução, caso ela exista. Quatro situações poderão ocorrer, e cada uma delas é descrita a seguir:

A solução obtida é ótima (linhas 8 a 13) – neste caso, o melhor limitante superior é atualizado, a região explorada é impedida de ser visitada novamente, com a adição da restrição $\Delta(x, \bar{x}) \geq rhs + 1$, e a variável $\bar{x}$ é atualizada com a nova solução $\tilde{x}$, em torno da qual a nova restrição de ramificação local será adicionada (linha 4). Caso a solução ótima tenha sido alcançada na primeira iteração, o algoritmo é encerrado (linha 10).

Não existe solução viável (linhas 14 a 21) – neste caso, não foi possível encontrar uma solução viável com valor inferior ao limitante informado. A região explorada é impedida de ser visitada novamente, com a adição da restrição $\Delta(x, \bar{x}) \geq rhs + 1$; a vizinhança a ser explorada passa a ser ampliada com o aumento de rhs e o parâmetro

³³ A presente implementação do “local branching” utiliza a solução inicial gerada pela heurística construtiva.

diversifica passa a permitir, em uma próxima iteração, a diversificação. Caso a solução se mostre inviável na primeira iteração, então o problema original não tem solução viável e o algoritmo é encerrado (linha 15).

Foi obtida solução viável (linhas 22 a 33) – neste caso, obteve-se uma solução melhor, mas o CPLEX não foi capaz de garantir a otimalidade naquele subespaço em função do limite de tempo, ou da imposição de interromper o processamento após a obtenção da primeira solução. No primeiro caso, uma nova restrição é incorporada $\Delta(x, \bar{x}) \geq 1$, de forma a manter ativo este subespaço de solução. No segundo caso, apenas a restrição de ramificação local, adicionada na linha 4, é removida. Seja qual for o caso, uma rotina de refinamento é chamada, que consiste em adicionar uma restrição de ramificação local $\Delta(x, \tilde{x}) = 0$, fixando todas as variáveis em $\tilde{x}$, para que se conheça, efetivamente, qual é a solução ótima deste subespaço.

Não foi encontrada solução viável (linhas 34 a 43) – neste caso, se o processo de diversificação estiver ativo, então os parâmetros UB, LT são desconsiderados (valores iguais a infinito), o tamanho da vizinhança é aumentado (*rhs*) e o CPLEX é configurado para rodar até que a primeira solução seja obtida. Caso o processo de diversificação não esteja configurado para ocorrer, então o tamanho da vizinhança é reduzido, de forma a concentrar a busca em uma região menor.

4.5 “Variable Neighborhood Search” (VNS)

A versão da busca em vizinhança variável implementada nesta pesquisa, para problemas MIP, é a proposta de Hansen et al. (2006) para a busca descendente em vizinhança variável. Conforme apresentado no capítulo 2, esta estratégia consiste em utilizar diversas estruturas de vizinhanças distintas e procurar pela melhor solução nestas vizinhanças, construídas em torno de uma solução corrente. O pseudocódigo

deste algoritmo é mostrado na figura 4.6. Os principais parâmetros e variáveis utilizados (maiores detalhes em Hansen et al.; 2006) estão indicados na tabela 4.5.

Tabela 4.5 - Lista de parâmetros e variáveis utilizados no VNS

Parâmetro / Variável	Descrição
f_atual	função objetivo da solução atual;
f_opt	função objetivo da melhor solução;
f_proximo	função objetivo da próxima solução a ser obtida;
k_passo	parâmetro que define o passo de crescimento da vizinhança;
k_atual	parâmetro que define o tamanho atual da vizinhança;
limite_tempo_nó	máximo tempo de processamento de um sub-problema;
LT	limite de tempo de processamento de um problema;
primeira_solução	indica se o algoritmo “branch and cut” deve ou não parar após encontrar a primeira solução;
rhs	parâmetro que define o tamanho da vizinhança;
stat	status da solução (ótima, inviável, viável, não encontrou);
tempo_total_processamento	máximo tempo de processamento do algoritmo VNS;
UB	limitante superior de um problema;
x_atual	solução atual de um problema;
x_opt	melhor solução já obtida;
x_proximo	próxima solução a ser obtida;

O algoritmo é inicializado na linha 1. Na linha 2, o comando ‘MIPSOLVE’ é chamado com o objetivo de gerar a solução inicial, que será posteriormente explorada (linhas 4 a 43). Para isso, o parâmetro *LT* é fixado no tempo máximo de processamento e *primeira_solução* igual a verdadeiro, fazendo com que a exploração da árvore “branch and bound” se encerre após a obtenção da primeira solução³⁴.

O loop principal do VNS (linhas 4 a 43) ocorre enquanto o tempo total de processamento não atingir o limite máximo. Dentro deste loop, um segundo bloco é repetido entre as linhas 6 e 22, e consiste na exploração da vizinhança centrada na solução corrente *x_atual*. Enquanto o status da solução for “encontrou solução ótima” (linhas 11 a 13) ou “encontrou solução viável” (linhas 14 a 16), significa que o valor

³⁴ A presente implementação do VNS utiliza a solução inicial gerada pela heurística construtiva.

da função objetivo está sendo melhorado. O valor de x_{atual} é atualizado e, em torno desta solução, uma nova vizinhança é construída com a adição de uma restrição de ramificação local (linha 8).

```

função VnsBra(tempo_total_processamento, limite_tempo_nó, k_passo, x_opt)
0 início
1   LT := tempo_total_processamento; UB := +∞; primeira_solução := verdadeiro;
2   stat := MIPSOLVE(LT, UB, primeira_solução, x_opt, f_opt)
3   x_atual := x_opt; f_atual := f_opt;
4   enquanto (tempo < tempo_total_processamento) fazer
5     cont := verdadeiro; rhs := 1; primeira_solução := falso;
6     enquanto (cont = verdadeiro e tempo < tempo_total_processamento) fazer
7       LT := mín( limite_tempo_nó, tempo_total_processamento - tempo);
8       adicionar restrição de ramificação local  $\Delta(x, x_{atual}) \leq rhs$ ; UB := f_atual;
9       stat := MIPSOLVE(LT, UB, primeira_solução, x_próximo, f_próximo)
10      fazer caso stat
11        = “encontrou solução ótima”
12          reverter última restrição de ramificação local para
13             $\Delta(x, x_{atual}) \geq rhs+1$ ;
14          x_atual := x_próximo; f_atual := f_próximo; rhs := 1;
15        = “encontrou solução viável”
16          reverter última restrição de ramificação local  $p/\Delta(x, x_{atual}) \geq 1$ ;
17          x_atual := x_próximo; f_atual := f_próximo; rhs := 1;
18        = “comprovada a inexistência de solução viável”
19          remover última restrição de ramificação local; rhs := rhs+1;
20        = “não encontrou solução viável”
21          cont := falso;
22      fim caso;
23    fim enquanto;
24    se (f_atual < f_opt) então
25      x_opt := x_atual; f_opt := f_atual; k_atual := k_passo;
26    senão
27      k_atual := k_atual + k_passo;
28    fim se;
29    remover todas as restrições de ramificação local adicionadas;
30    cont := verdadeiro;
31    enquanto (cont = verdadeiro e tempo < tempo_total_processamento) fazer
32      adicionar restrição  $k_{atual} \leq \Delta(x, x_{opt}) \leq k_{atual} + k_{passo}$ 
33      LT := tempo_total_processamento - tempo; UB := +∞
34      primeira_solução := verdadeiro;
35      stat := MIPSOLVE(LT, UB, primeira_solução, x_atual, f_atual)
36      remover duas restrições de ramificação local adicionadas;
37      cont := falso;
38      se (stat = “não encontrou solução viável” ou
39        stat = “comprovada a inexistência de solução viável”) então
40        cont := verdadeiro; k_atual := k_atual + k_passo;
41      fim se;
42    fim enquanto;
43  fim enquanto;
44 fim;

```

Figura 4.6 – Pseudo código do algoritmo “VNS” (adaptado de Hansen, et al., 2006)

Se o status da solução for “comprovada a inexistência de solução ótima”, significa que não foi possível obter solução inferior ao limitante superior UB naquela região. Neste caso, o parâmetro de definição da vizinhança rhs é aumentado.

Apenas se nenhuma solução for encontrada dentro do tempo limite de processamento ($limite_tempo_nó$), o loop é interrompido. Nesse caso, todas as restrições de ramificação local são removidas (linha 28) e uma nova vizinhança é definida em torno da solução ótima x_{opt} (linha 31). As imposições de limite de tempo (LT) e limitante superior (UB) são relaxadas (linha 32) e o algoritmo “branch and cut” é executado até que a primeira solução seja obtida (linhas 30 a 42).

4.6 Conclusão

Este capítulo teve por finalidade apresentar os métodos de solução utilizados nesta pesquisa, entre eles: a heurística construtiva, em duas versões, o método “feasibility pump” e os métodos “local branching” e “variable neighborhood search”. Cabe lembrar que os últimos três métodos irão interagir diretamente com o software de otimização matemática CPLEX 10.0.

Os problemas usados para testes e os detalhes de cada processamento, tais como a duração e a calibração dos parâmetros relevantes, serão abordados no próximo capítulo.

5 APLICAÇÃO DOS MÉTODOS DE SOLUÇÃO

Este capítulo apresenta os resultados da aplicação dos métodos de solução ao problema de PROGRAMAÇÃO DE FROTA DE APOIO A OPERAÇÕES “OFFSHORE” SUJEITA À REQUISICÃO DE MÚLTIPLAS EMBARCAÇÕES PARA UMA MESMA TAREFA. Primeiramente será introduzida a base de problemas utilizada e em seguida os resultados obtidos pela aplicação de cada método aos problemas serão apresentados e discutidos.

Os métodos de solução descritos no capítulo anterior constituem estratégias importantes para a resolução do problema desta pesquisa. Contudo, pretende-se examinar a eficácia destes métodos isoladamente, como também combinar estas estratégias de forma a acelerar o processo de busca e, eventualmente, de convergência da solução.

Além de aplicar o algoritmo clássico “branch and cut” ao conjunto de problemas, será avaliada a aplicação deste algoritmo tendo como ponto de partida as soluções geradas pelas heurísticas construtivas. O mesmo será feito com os métodos “local branching” e “variable neighborhood search”, os quais utilizarão soluções iniciais geradas por estas heurísticas. Em seguida, visando aprimorar o processo de geração de soluções iniciais, será testada a estratégia “feasibility pump”. A tabela 5.1 lista os métodos que serão aplicados e a quais modelos matemáticos, e indica se cada método será testado com base em uma solução inicial ou não.

O primeiro e o segundo métodos serão a aplicação do algoritmo “branch and cut” aos modelos de fluxo em rede nas suas versões completa e compacta, respectivamente. O terceiro e o quarto métodos referem-se à aplicação das heurísticas

construtivas nas versões básica e gulosa, respectivamente. Em seguida, a melhor solução heurística obtida para cada cenário (ou, problema) servirá de solução inicial para a aplicação dos métodos “branch and cut”, “local branching” e “variable neighborhood search”, sendo estes os métodos de solução 5, 6 e 7, respectivamente. O método “branch and cut” também será aplicado na resolução do modelo de programação de frota em escala de tempo discretizada, sem e com deslocamento, sendo estas as estratégias 8 e 9. Por último, serão conduzidos testes com o método “feasibility pump” aplicado ao modelo de fluxo em rede – versão compacta, e com o modelo de programação de frota em escala de tempo discretizada, sem deslocamento.

Tabela 5.1 - Métodos de Solução e Correspondetes Modelos Matemáticos

Método	Algoritmo	Solução Inicial	Modelo
1	Branch and cut		Fluxo em Rede - Completo
2	Branch and cut		Fluxo em Rede - Compacto
3	Heurística construtiva - versão básica		Fluxo em Rede - Completo
4	Heurística construtiva - versão gulosa		Fluxo em Rede - Completo
5	Branch and cut	Melhor(HC Básica; HC Gulosa)	Fluxo em Rede - Compacto
6	VNS	Melhor(HC Básica; HC Gulosa)	Fluxo em Rede - Compacto
7	Local Branching	Melhor(HC Básica; HC Gulosa)	Fluxo em Rede - Compacto
8	Branch and cut		Programação Frota em Escala Tempo Discretizada sem Deslocamento – II
9	Branch and cut		Programação Frota em Escala Tempo Discretizada com Deslocamento
10	Feasibility Pump		Fluxo em Rede – Compacto
11	Feasibility Pump		Programação Frota em Escala Tempo Discretizada sem Deslocamento – II

Dentre os modelos apresentados no capítulo 3, dois deles não serão testados. Um deles é o modelo de “fluxo em rede – conjuntos”, pois o número de conjuntos de embarcações presentes nos problemas testados introduziu uma carga adicional significativa de variáveis binárias. Outro modelo que não será testado é o modelo “programação de frota em escala tempo discretizada, sem deslocamento - versão I”, pois o desempenho deste modelo, em testes preliminares, foi inferior ao da versão II.

Para comparar a eficácia dos métodos, isto é, suas capacidades de gerar soluções de alta qualidade em um determinado período, fixou-se um tempo limite de processamento computacional igual a doze horas. Esta duração foi definida em função do porte dos problemas resolvidos e dos recursos computacionais disponíveis para a realização desta pesquisa. Assim, os resultados obtidos pelos processamentos do método “branch and cut”, limitados a doze horas de duração (métodos 1 e 2) serão comparados com os resultados dos métodos combinados 5, 6 e 7, em que as heurísticas construtivas serão processadas por 6 horas e os métodos de melhoria, aplicados subsequenteemente (“branch and cut”, “vns” e “local branching”), também terão um tempo máximo de processamento igual a seis horas. A heurística “feasibility pump” também será processada por 6 horas, com o intuito de verificar sua eficácia na geração de soluções iniciais.

5.1 Base de Dados

O tema desta pesquisa foi motivado pelo conhecimento de um problema real de programação de embarcações de apoio a operações “offshore”. O contato com a empresa de petróleo possibilitou o acesso a uma base de dados de demanda representativa do problema em questão. Devido aos aspectos de sigilo dos dados e, principalmente, pela necessidade de uma base mais diversificada, que permitisse testar a robustez dos algoritmos desenvolvidos, optou-se pelo desenvolvimento de

uma base de dados fictícia, gerada por um procedimento probabilístico, conservando as características do problema real.

Para isso foi criado um gerador de cenários capaz de construir cenários representativos da operação, requerendo como parâmetros de entrada:

- *Número de tarefas a serem criadas;*
- *Horizonte de planejamento* – extensão ou período total de planejamento em dias, dentro do qual as tarefas serão distribuídas;
- *Duração das tarefas* – duração mínima e máxima das tarefas em dias;
- *Extensão da janela de tempo* – período máximo durante o qual uma tarefa pode ter início;
- *Penalidade* – valor mínimo e máximo da penalidade por atraso no início de uma tarefa [\$/dia];
- *Coordenadas* – cada tarefa possui coordenadas referentes aos locais de início e término da atividade que podem, eventualmente, ser distintas. A representação do problema requer a definição da faixa de coordenadas válidas para geração dos diversos locais onde ocorrerão as tarefas. Apesar de o problema real usualmente utilizar o sistema de coordenadas geodésicas, simplificou-se esta representação, introduzindo-se o sistema de coordenadas cartesianas;
- *Locais de Início e Término* – após os locais candidatos terem sido gerados, são atribuídos a cada tarefa locais de início e término;
- *Tarefa predecessora* – pode-se impor que algumas tarefas tenham tarefas predecessoras. Nesse caso, tarefas são sorteadas e a relação de precedência entre elas é imposta;
- *Embarcações por Classe* – o último parâmetro a ser informado é o número mínimo e máximo de embarcações por classe que comporão a demanda de cada tarefa.

Cada cenário ou instância é composto pelo conjunto de tarefas e pelo conjunto de embarcações. Para gerar cada tarefa de um cenário devem ser usados os valores

dos parâmetros descritos. Alguns parâmetros, como o instante de início de uma tarefa, a sua duração, a extensão da janela de tempo, o número de embarcações requisitadas por classe, e os locais de início e término da tarefa são discretos, e suas definições ocorrem mediante o sorteio equiprovável dentre a faixa de valores possíveis. Para outros parâmetros, como a penalidade e as coordenadas que definem os locais das tarefas, utiliza-se uma função densidade de probabilidade de uma distribuição uniforme.

Foram criados, ao todo, 48 problemas igualmente distribuídos entre os cenários de 10, 20, 30 e 40 tarefas. A tabela 5.2 indica os parâmetros e os respectivos valores empregados no processo de criação da base de dados. As colunas desta tabela referem-se a: o número do cenário; o número de tarefas por cenário (N_t); o número de embarcações do cenário (N_e); a extensão do horizonte de planejamento (H_p); a duração média das tarefas (p_m); o número de dias em que uma tarefa pode ter o seu início antecipado ou atrasado, definindo, assim, sua janela de tempo; o número médio de embarcações por tarefa (N_{et}); e a ocupação nominal das embarcações. Este último parâmetro é calculado conforme mostra a equação 5.1, e fornece um indicador de ocupação da frota considerando apenas os tempos de execução das tarefas, excluindo os tempos de deslocamento das embarcações.

$$\text{ocupação} = \frac{N_t p_m N_{et}}{N_e H_p} \quad (5.1)$$

Outros parâmetros utilizados foram: o número de classes de embarcações, fixo em 2; a faixa para sorteio das durações das tarefas, variando entre 2 e 18 dias; a faixa para sorteio das penalidades diárias, variando entre R\$ 0,00 e R\$ 320.000,00; a probabilidade de um local de início ser o mesmo do fim da tarefa, igual a 80%; e a faixa para sorteio das coordenadas x e y, dos locais das tarefas, variando entre 0 e 300 milhas náuticas.

Tabela 5.2 – Parâmetros dos cenários

Cenário	# Tarefas	# Embarcações	Horizonte Planej. [dias]	Duração Média [dias]	Antecip/Atraso [dias]	Média Emb. por Tarefa	Ocup. Nominal Emb.
1	10	12	30	10,0	5	2,4	66,0%
2	10	12	30	10,0	5	2,6	71,5%
3	10	13	30	10,0	5	2,8	72,5%
4	10	13	30	12,0	5	2,4	73,2%
5	10	12	30	12,0	5	2,6	87,4%
6	10	16	30	12,0	5	2,8	70,0%
7	10	11	30	10,0	7	2,4	73,5%
8	10	12	30	10,0	7	2,6	71,5%
9	10	13	30	10,0	7	2,8	72,5%
10	10	12	30	12,0	7	2,4	80,0%
11	10	14	30	12,0	7	2,6	74,3%
12	10	15	30	12,0	7	2,8	74,7%
13	20	22	30	10,0	5	2,4	72,0%
14	20	23	30	10,0	5	2,6	74,6%
15	20	27	30	10,0	5	2,8	68,4%
16	20	27	30	12,0	5	2,4	70,8%
17	20	29	30	12,0	5	2,6	71,4%
18	20	31	30	12,0	5	2,8	72,6%
19	20	22	30	10,0	7	2,4	72,7%
20	20	23	30	10,0	7	2,6	71,6%
21	20	27	30	10,0	7	2,8	62,9%
22	20	25	30	12,0	7	2,4	76,8%
23	20	29	30	12,0	7	2,6	72,0%
24	20	31	30	12,0	7	2,8	72,3%
25	30	23	45	10,0	5	2,4	68,9%
26	30	26	45	10,0	5	2,6	67,3%
27	30	27	45	10,0	5	2,8	69,6%
28	30	28	45	12,0	5	2,4	68,4%
29	30	29	45	12,0	5	2,6	71,9%
30	30	31	45	12,0	5	2,8	72,9%
31	30	21	45	10,0	7	2,4	75,9%
32	30	23	45	10,0	7	2,6	75,4%
33	30	26	45	10,0	7	2,8	72,0%
34	30	25	45	12,0	7	2,4	77,4%
35	30	29	45	12,0	7	2,6	71,9%
36	30	31	45	12,0	7	2,8	72,3%
37	40	22	60	10,0	5	2,4	71,8%
38	40	25	60	10,0	5	2,6	69,5%
39	40	27	60	10,0	5	2,8	68,1%
40	40	26	60	12,0	5	2,4	73,8%
41	40	27	60	12,0	5	2,6	76,4%
42	40	32	60	12,0	5	2,8	70,9%
43	40	23	60	10,0	7	2,4	70,3%
44	40	25	60	10,0	7	2,6	68,5%
45	40	27	60	10,0	7	2,8	68,6%
46	40	27	60	12,0	7	2,4	70,5%
47	40	27	60	12,0	7	2,6	77,8%
48	40	32	60	12,0	7	2,8	69,9%

5.2 “Branch & cut” aplicado ao Modelo de Fluxo em Rede – Completo (BC Flx Completo)

O algoritmo “branch & cut” que acompanha o pacote computacional CPLEX 10.0 foi aplicado a este conjunto de 48 problemas, usando, inicialmente, a formulação matemática do Modelo de Fluxo em Rede – Completo. Cada problema foi processado por um período de 12 horas em um computador Pentium 4, 2.4 GHz, 512 Mb RAM, HD local PATA 40 Gb. Os resultados estão apresentados na tabela 5.3.

A tabela 5.3 lista, para cada cenário, o número de tarefas, o tempo total de processamento (limitado a 12 horas, ou 43.200 segundos), o número total de variáveis, o número de variáveis binárias e de restrições presentes no modelo, o número de soluções geradas pelo método “branch and cut”, o valor da função objetivo e a melhor estimativa de limitante inferior para a função objetivo, o número de nós da árvore de solução gerada pelo método “branch and bound” e o “gap” da solução.

Além do critério de parada por tempo (12 horas), há interrupção do processamento quando o “gap”, definido como a distância relativa da melhor estimativa da função objetivo (“*bound*”) em relação à função objetivo da melhor solução obtida ($\tilde{f}_{opt}$) for menor ou igual a 1%: $gap = (\tilde{f}_{opt} - bound) / \tilde{f}_{opt} \leq 1\%$.

Cabe mencionar que, conforme o método de solução empregado (1 a 11 da tabela 5.1), diferentes limitantes inferiores foram sendo fornecidos pelo pacote de otimização CPLEX. Adotou-se como melhor estimativa da função objetivo (“*bound*”) o maior limitante obtido nos diversos processamentos que foram realizados com cada cenário, de forma a padronizar a comparação de desempenho entre os métodos.

Tabela 5.3 – Resultados do método “branch and cut” aplicado ao modelo de fluxo em rede – formulação completa

Cenário	Tarefa	Tempo[s]	#Var.	#Bin.	#Restr.	#Sol.	F Obj. [\$]	Bound [\$]	Nós	GAP
1	10	657,39	2.684	1.332	4.188	16	221.478,92	219.423,69	9.470	0,9%
2	10	93,63	2.684	1.332	4.188	8	320.903,43	320.012,59	807	0,3%
3	10	3.330,40	2.906	1.443	4.532	16	121.749,50	120.624,70	23.015	0,9%
4	10	6,33	2.906	1.443	4.532	5	305.256,69	302.305,30	70	1,0%
5	10	160,38	2.684	1.332	4.188	12	979.990,95	977.786,19	1.337	0,2%
6	10	115,44	3.572	1.776	5.564	17	182.949,79	182.375,34	1.101	0,3%
7	10	19.459,80	2.462	1.221	3.844	7	402.002,50	397.989,15	115.000	1,0%
8	10	26.115,00	2.684	1.332	4.188	14	629.357,48	623.207,22	119.491	1,0%
9	10	103,74	2.906	1.443	4.532	6	237.501,42	235.883,06	773	0,7%
10	10	43.200,00	2.684	1.332	4.188	19	208.245,72	110.569,31	182.176	46,9%
11	10	25,67	3.128	1.554	4.876	9	1.009.664,51	1.005.869,64	210	0,4%
12	10	30.242,90	3.350	1.665	5.220	25	261.365,23	258.981,11	158.239	0,9%
13	20	43.200,00	18.564	9.262	28.368	11	528.829,60	103.460,71	20.955	80,4%
14	20	43.200,00	19.406	9.683	29.652	13	1.043.208,27	104.600,50	22.302	90,0%
15	20	43.200,00	22.774	11.367	34.788	19	151.159,12	100.798,86	25.651	33,3%
16	20	43.200,00	22.774	11.367	34.788	39	1.986.652,01	836.356,58	35.782	57,9%
17	20	43.200,00	24.458	12.209	37.356	18	1.575.170,98	409.679,92	22.479	74,0%
18	20	43.200,00	26.142	13.051	39.924	18	1.258.439,29	116.147,01	22.553	90,8%
19	20	43.200,00	18.564	9.262	28.368	7	715.186,72	91.124,48	15.738	87,3%
20	20	43.200,00	19.406	9.683	29.652	4	2.457.835,57	79.218,05	11.174	96,8%
21	20	43.200,00	22.774	11.367	34.788	4	5.578.286,28	83.966,34	10.212	98,5%
22	20	43.200,00	21.090	10.525	32.220	5	1.361.556,70	92.179,24	9.444	93,2%
23	20	43.200,00	24.458	12.209	37.356	11	1.643.079,91	132.499,32	16.991	91,9%
24	20	43.200,00	26.142	13.051	39.924	17	1.029.994,56	112.644,03	14.032	89,1%
25	30	43.200,00	42.886	21.413	65.132	0		92.262,69		
26	30	43.200,00	48.472	24.206	73.604	5	1.762.501,93	1.585.734,27	7.685	10,0%
27	30	43.200,00	50.334	25.137	76.428	3	1.570.679,79	113.644,97	7.844	92,8%
28	30	43.200,00	52.196	26.068	79.252	8	272.703,36	133.215,49	8.807	51,2%
29	30	43.200,00	54.058	26.999	82.076	6	1.003.693,14	113.395,78	7.086	88,7%
30	30	43.200,00	57.782	28.861	87.724	10	1.297.483,36	138.425,94	7.759	89,3%
31	30	43.200,00	39.162	19.551	59.484	2	3.425.403,80	83.382,64	5.767	97,6%
32	30	43.200,00	42.886	21.413	65.132	0		79.391,98		
33	30	43.200,00	48.472	24.206	73.604	0		96.186,69		
34	30	43.200,00	46.610	23.275	70.780	5	4.309.820,86	106.465,54	5.602	97,5%
35	30	43.200,00	54.058	26.999	82.076	0		203.516,78		
36	30	43.200,00	57.782	28.861	87.724	0		115.302,92		
37	40	43.200,00	72.284	36.102	109.448	0		837.441,35		
38	40	43.200,00	82.130	41.025	124.340	0		119.359,60		
39	40	43.200,00	88.694	44.307	134.268	3	11.871.173,16	137.477,49	2.611	98,8%
40	40	43.200,00	85.412	42.666	129.304	0		344.488,74		
41	40	43.200,00	88.694	44.307	134.268	0		133.939,73		
42	40	43.200,00	105.104	52.512	159.088	8	1.439.097,26	124.181,63	2.771	91,4%
43	40	43.200,00	75.566	37.743	114.412	5	10.005.453,44	104.870,47	2.526	99,0%
44	40	43.200,00	82.130	41.025	124.340	0		115.798,42		
45	40	43.200,00	88.694	44.307	134.268	0		102.533,83		
46	40	43.200,00	85.412	42.666	129.304	0		118.747,22		
47	40	43.200,00	88.694	44.307	134.268	0		196.218,86		
48	40	43.200,00	105.104	52.512	159.088	0		124.117,00		

A análise dos dados da tabela 5.3 indica que em 34 dos 48 problemas (71%) obteve-se pelo menos uma solução viável, sendo que em 11 problemas a solução obtida foi a ótima, segundo o critério de otimalidade (“gap” $\leq 1\%$). O valor médio do “gap” para os 34 problemas em que se obteve pelo menos uma solução viável foi 54,5%.

A tabela 5.4 agrupa os resultados da tabela 5.3 por quantidade de tarefas, indicando para quantos problemas pelo menos uma solução viável foi gerada e quantas destas soluções são ótimas. A tabela também mostra os valores médios do número total de variáveis, de variáveis binárias, de restrições, o número médio de soluções e a quantidade média de nós da árvore “branch and cut” gerados no decorrer do processamento, e o valor médio do “gap”. A incapacidade de resolver problemas de porte maior é comprovada pelo baixo número de soluções geradas durante o período de 12 horas (ver coluna # Médio Soluções Geradas), sendo esta decorrente do baixo número de nós explorados pelo algoritmo “branch and cut” (ver coluna # Médio Nós).

Tabela 5.4 – Resultados agrupados por número de tarefas

# Tarefas	# Problemas c/ Solução Inteira	# Problemas c/ Solução Ótima	# Médio Variáveis	# Médio Variáveis Binárias	# Médio Restrições	# Médio Soluções Geradas	# Médio Nós	GAP Médio
10	12	11	2.888,0	1.434,0	4.503,0	12,8	50.974,0	4,50%
20	12	0	22.213,0	11.086,0	33.932,0	13,8	18.943,0	81,90%
30	7	0	49.558,0	24.749,0	75.251,0	5,6	7.221,0	75,30%
40	3	0	87.327,0	43.623,0	132.200,0	5,3	2.636,0	96,40%

5.3 “Branch & cut” aplicado ao Modelo de Fluxo em Rede – Compacto (BC Flx Compacto)

A versão compacta dos modelos de fluxo em rede foi, em seguida, testada com o método “branch and cut”. Assim como na versão completa, cada problema foi processado por um período de 12 horas, tendo como critério de parada o “gap” menor ou igual a 1%. A tabela 5.5 lista os resultados obtidos, possuindo a mesma estrutura da tabela 5.3.

A análise dos dados da tabela 5.5 indica que em 38 dos 48 problemas (79%) obteve-se pelo menos uma solução viável, sendo que em 11 problemas a solução obtida foi a ótima, segundo o critério de parada (“gap” $\leq 1\%$). O valor médio do “gap” das 38 soluções geradas foi de 49,6%.

Comparando com desempenho do “branch and cut” aplicado à formulação completa de fluxo em rede, apenas para o problema 43 a formulação completa foi capaz de obter uma solução inteira, enquanto que a formulação compacta não conseguiu. Os cenários que foram resolvidos apenas com a formulação compacta, isto é, a formulação completa não foi capaz de resolver, são os 25, 35, 37, 40 e 45. Já os cenários 32, 33, 36, 38, 41, 44, 46, 47 e 48, não foram resolvidos por nenhum dos dois modelos.

A tabela 5.6 agrupa os resultados da tabela 5.5 por quantidade de tarefas, indicando para quantos problemas pelo menos uma solução viável foi gerada e quantas destas soluções são ótimas. A tabela também mostra os valores médios do número total de variáveis, de variáveis binárias, de restrições, o número médio de soluções e a quantidade média de nós da árvore “branch and cut” gerados no decorrer do processamento, e o valor médio do “gap”.

Tabela 5.5 - Resultados do método “branch and cut” aplicado ao modelo de fluxo em rede – formulação compacta

Cenário	Tarefa	Tempo[s]	#Var.	#Bin.	#Restr.	#Sol.	F.Obj. [\$]	Bound [\$]	Nós	GAP
1	10	27,64	1.460	720	1.512	8	221.478,92	219.423,69	1.280	0,9%
2	10	4,85	1.508	744	1.560	6	320.918,10	320.012,59	147	0,3%
3	10	236,74	1.612	796	1.665	16	121.752,49	120.624,70	6.403	0,9%
4	10	2,65	1.424	702	1.477	7	304.837,97	302.305,30	100	0,8%
5	10	18,14	1.268	624	1.320	6	980.589,10	977.786,19	447	0,3%
6	10	11,37	1.716	848	1.772	7	182.930,96	182.375,34	313	0,3%
7	10	2.741,74	1.604	792	1.655	11	401.906,02	397.989,15	41.571	1,0%
8	10	2.808,54	1.892	936	1.944	9	629.357,48	623.207,22	46.312	1,0%
9	10	37,47	1.736	858	1.789	4	238.231,94	235.883,06	793	1,0%
10	10	43.200,00	1.556	768	1.608	26	207.867,45	110.569,31	590.100	46,8%
11	10	10,04	1.728	854	1.782	8	1.012.575,67	1.005.869,64	320	0,7%
12	10	1.188,83	1.850	915	1.905	13	261.365,23	258.981,11	35.809	0,9%
13	20	43.200,20	9.228	4.594	9.330	17	629.656,66	103.460,71	52.622	83,6%
14	20	43.200,10	10.160	5.060	10.263	19	909.638,55	104.600,50	43.173	88,5%
15	20	43.200,10	11.556	5.758	11.663	26	107.582,21	100.798,86	55.548	6,3%
16	20	43.200,10	8.318	4.139	8.425	48	1.827.501,26	836.356,58	83.482	54,2%
17	20	43.200,10	10.008	4.984	10.117	32	556.027,95	409.679,92	94.890	26,3%
18	20	43.200,10	11.076	5.518	11.187	20	742.715,71	116.147,01	51.185	84,4%
19	20	43.200,20	11.876	5.918	11.978	10	1.203.817,12	91.124,48	36.821	92,4%
20	20	43.200,20	12.920	6.440	13.023	11	1.468.270,15	79.218,05	22.361	94,6%
21	20	43.200,30	17.442	8.701	17.549	5	4.107.550,39	83.966,34	16.790	98,0%
22	20	43.200,20	12.932	6.446	13.037	19	1.662.005,81	92.179,24	23.408	94,5%
23	20	43.200,10	12.240	6.100	12.349	30	444.872,75	132.499,32	33.145	70,2%
24	20	43.200,10	14.700	7.330	14.811	13	807.196,86	112.644,03	24.462	86,0%
25	30	43.200,40	21.864	10.902	22.007	9	1.891.799,93	92.262,69	18.697	95,1%
26	30	43.200,40	24.292	12.116	24.438	29	1.660.976,06	1.585.734,27	29.402	4,5%
27	30	43.200,20	26.034	12.987	26.181	9	162.732,27	113.644,97	21.236	30,2%
28	30	43.200,20	24.270	12.105	24.418	7	863.941,85	133.215,49	24.889	84,6%
29	30	43.200,40	25.116	12.528	25.265	7	129.730,26	113.395,78	23.539	12,6%
30	30	43.200,20	26.348	13.144	26.499	17	142.818,37	138.425,94	30.312	3,1%
31	30	43.200,30	23.790	11.865	23.931	1	5.589.106,09	83.382,64	13.980	98,5%
32	30	43.200,00	27.588	13.764	27.731	0		79.391,98		
33	30	43.200,00	29.960	14.950	30.106	0		96.186,69		
34	30	43.200,00	26.052	12.996	26.197	4	2.007.944,88	106.465,54	14.149	94,7%
35	30	43.200,30	30.012	14.976	30.161	3	7.388.233,59	203.516,78	9.376	97,2%
36	30	43.200,00	33.106	16.523	33.257	0		115.302,92		
37	40	43.200,50	37.040	18.480	37.222	3	2.400.857,11	837.441,35	12.164	65,1%
38	40	43.200,00	41.980	20.950	42.165	0		119.359,60		
39	40	43.200,40	45.708	22.814	45.895	5	5.245.226,34	137.477,49	7.980	97,4%
40	40	43.200,50	40.824	20.372	41.010	19	2.572.777,40	344.226,82	13.760	86,6%
41	40	43.200,00	42.678	21.299	42.865	0		133.939,73		
42	40	43.200,60	50.030	24.975	50.222	6	598.289,07	124.181,63	10.099	79,2%
43	40	43.200,00	43.458	21.689	43.641	0		104.870,47		
44	40	43.200,00	46.202	23.061	46.387	0		115.798,42		
45	40	43.200,60	51.272	25.596	51.459	4	1.763.845,32	102.213,20	5.219	94,2%
46	40	43.200,00	47.762	23.841	47.949	0		118.747,22		
47	40	43.200,00	46.828	23.374	47.015	0		196.218,86		
48	40	43.200,00	56.428	28.174	56.620	0		124.117,00		

Tabela 5.6 - Resultados agrupados por número de tarefas

# Tarefas	# Problemas c/ Solução Inteira	# Problemas c/ Solução Ótima	# Médio Variáveis	# Médio Variáveis Binárias	# Médio Restrições	# Médio Soluções Geradas	# Médio Nós	GAP Médio
10	12	11	1.612,8	796,4	1.665,8	10,1	60.299,6	4,57%
20	12	0	11.871,3	5.915,7	11.977,7	20,8	44.983,3	73,31%
30	7	0	26.463,7	13.201,8	26.610,2	7,2	20.620,0	57,84%
40	6	0	45.850,8	22.885,4	46.037,5	3,1	9.844,4	84,59%

A eficiência da formulação compacta é refletida por um “gap” médio inferior, por classe de problemas (10, 20, 30 e 40 tarefas), e por uma quantidade maior de problemas resolvidos (quatro a mais), quando comparado com os resultados da formulação completa. Isto se justifica por alguns elementos, como o número de variáveis binárias, que é, em média, 47% menor que na formulação completa. O número de restrições também caiu, em média, 65% em relação à formulação completa. Com uma estrutura mais compacta, o número de nós da árvore “branch and cut” explorados aumentou, em média, 18,3%, 137,5%, 194,5% e 273,5% para as classes de 10, 20, 30 e 40 tarefas, respectivamente. Além disso, para os 11 problemas de 10 tarefas, para os quais a solução ótima foi obtida, o tempo de processamento computacional usando a formulação compacta foi, em média, 83,3% menor.

A figura 5.1 exemplifica a evolução do número de soluções geradas e dos valores da função objetivo para o cenário 20, ressaltando o aspecto de maior rapidez na exploração dos nós da árvore “branch and cut”, para a formulação compacta.

Uma solução gerada pelo método “branch and cut” pode ser interpretada por meio da análise das variáveis binárias de decisão que compõem a solução do problema. A forma como uma solução pode ser estruturada e compreendida será exemplificada para o cenário 5. Neste problema, a demanda consiste em um conjunto de 10 tarefas, cujas características estão descritas na tabela 5.7. As colunas mostram, para cada tarefa, os locais de início e término de sua execução (dentro um conjunto

de locais possíveis, listados na tabela 5.9), os limites inferior e superior da janela de tempo, o instante desejado de início, a partir do qual incide a penalidade por atraso, a duração da tarefa, em dias, a penalidade diária e a demanda por classe de embarcações.

Figura 5.1 – Comparação do desempenho do método “branch and cut” para o cenário #20, aplicando-o às formulações completa e compacta do modelo de fluxo em rede

Tabela 5.7 – Dados das tarefas – cenário #5

Tarefa	Local Início	Local Término	Limite Inferior	Limite Superior	Data Desejada	Duração [dia]	Penalidade Atraso [\$/dia]	Demanda Classe 1	Demanda Classe 2
1	14	14	12/1/2006	22/1/2006	17/1/2006	11	118.760,00	2	2
2	20	20	1/1/2006	8/1/2006	3/1/2006	13	303.050,00	3	0
3	15	15	1/1/2006	8/1/2006	3/1/2006	17	94.459,00	1	1
4	1	2	7/1/2006	17/1/2006	12/1/2006	6	227.398,00	2	1
5	16	16	1/1/2006	11/1/2006	6/1/2006	17	7.078,00	2	0
6	11	11	4/1/2006	14/1/2006	9/1/2006	13	192.547,00	1	1
7	16	16	15/1/2006	25/1/2006	20/1/2006	10	242.608,00	2	1
8	7	7	1/1/2006	11/1/2006	6/1/2006	20	93.031,00	1	0
9	10	10	1/1/2006	9/1/2006	4/1/2006	3	199.945,00	2	1
10	6	6	12/1/2006	22/1/2006	17/1/2006	11	4.652,00	3	0

Os dados referentes às embarcações encontram-se na tabela 5.8. Para cada embarcação, a tabela contém a classe a que pertence, a velocidade, em nós, o custo variável diário, os instantes de início e término do contrato de afretamento, a autonomia da embarcação, o tempo gasto no abastecimento, o local e o instante em que a embarcação está disponível para a operação.

A solução gerada para este cenário, por meio do método “branch and cut”, indica os fluxos de embarcações entre cada par de locais (“origem” e “destino”) que foram escolhidos. Os locais “0” e “11” representam as tarefas fictícias, isto é, as condições inicial e final das embarcações (neste exemplo, a base), conforme pode ser visto na tabela 5.10. Cada um dos 38 ternos (embarcação, origem, destino) desta tabela corresponde a uma variável binária x_{ij}^r que assumiu valor 1 na solução do modelo, por exemplo, $x_{0-5}^1 = 1$; $x_{5-7}^1 = 1$; $x_{7-11}^1 = 1$, e assim sucessivamente.

Tabela 5.8 – Dados das embarcações – cenário #5

Embarcação	Classe	Veloc. [nós]	Custo Variável [\$ / dia]	Início Contrato	Término Contrato	Autonomia [dia]	Tempo Abast. [dia]	Local Disp.	Instante Disp.
1	1	12,0	2.000,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
2	1	12,0	2.000,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
3	1	12,0	2.000,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
4	1	12,0	2.000,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
5	1	12,5	2.125,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
6	1	12,5	2.125,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
7	1	12,5	2.125,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
8	1	12,5	2.125,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
9	2	14,0	3.000,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
10	2	14,5	3.175,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
11	2	14,7	3.245,00	1/1/2006	1/3/2006	60	1	0	1/1/2006
12	2	14,7	3.245,00	1/1/2006	1/3/2006	60	1	0	1/1/2006

Tabela 5.9 – Dados dos locais – cenário #5

Local	X [milhas]	Y [milhas]	Local	X [milhas]	Y [milhas]
0	0	0	11	51	177
1	215	84	12	152	278
2	224	89	13	28	146
3	86	164	14	11	269
4	210	44	15	220	51
5	3	290	16	132	177
6	270	123	17	60	107
7	267	185	18	109	184
8	116	269	19	1	175
9	255	222	20	6	265
10	69	284			

Tabela 5.10 – Resultados do método “branch and cut” – cenário #5

Emb.	Origem	Destino	Emb.	Origem	Destino	Emb.	Origem	Destino
1	0	5	5	0	2	9	0	3
1	5	7	5	2	4	9	3	7
1	7	11	5	4	11	9	7	11
2	0	5	6	0	2	10	0	9
2	5	7	6	1	11	10	4	10
2	7	11	6	2	1	10	9	4
3	0	9	7	0	2	10	10	11
3	8	11	7	2	4	11	0	6
3	9	8	7	4	10	11	1	11
4	0	9	7	10	11	11	6	1
4	1	11	8	0	3	12	0	6
4	9	1	8	3	10	12	1	11
			8	10	11	12	6	1

Embora a especificação da solução, como mostrada na tabela 5.10, não contenha os instantes de início e término de viagem das embarcações, nem os instantes em que as tarefas têm início, o valor destas variáveis pode ser calculado a partir das variáveis de fluxo da solução. Para isso, basta somar aos instantes de disponibilidade inicial das embarcações os tempos de viagem de cada uma delas para os seus respectivos locais de destino. Cada tarefa terá início apenas quando a última embarcação a ela designada tiver chegado, e se este instante for superior ao limite inferior da janela de tempo. Após a conclusão de uma tarefa, cada embarcação irá se deslocar para o local de início da próxima tarefa, ou para a base.

A tabela 5.11 resume este procedimento. Para cada embarcação, a tabela especifica a velocidade média, os locais de origem e destino, a distância entre estes locais, o tempo de viagem, a parcela de custo variável decorrente desta viagem e o instante de chegada. Em seguida, são mostrados a duração da tarefa, o limite inferior da janela de tempo para início da tarefa e o início efetivo da mesma. Este é calculado como o máximo instante de chegada dentre as embarcações designadas a esta tarefa, caso este instante seja maior ao limite inferior da janela de tempo. Em caso contrário, as embarcações deverão aguardar até o limite inferior. Por último, tem-se o instante de liberação das embarcações, que é dado pela soma do instante de início mais a duração da tarefa.

Por exemplo, a embarcação 1 chega ao local de início da tarefa 5 no instante 1,77. Como esta tarefa demanda duas embarcações, é necessário aguardar o instante em que a outra embarcação designada para esta tarefa chegará a esse local. Por ter a segunda embarcação características semelhantes (mesma velocidade) à primeira e por partirem ambas da base no instante inicial ($t=1$), a chegada da segunda embarcação também ocorrerá no instante 1,77, quando a tarefa será iniciada. Eventualmente, as embarcações terão que aguardar até que o limite inferior da janela de tempo seja atingido. No instante 18,77 estas embarcações estarão liberadas, quando poderão ser designadas a alguma outra tarefa.

O instante de início de uma tarefa depende, portanto, da chegada ao local de início de execução de todas as embarcações a ela designadas e, assim, pode, eventualmente, ocorrer atraso para o início da tarefa. A tabela 5.12 mostra, para cada tarefa, o instante de início e se houve atraso em relação à data desejada de início.

Tabela 5.11 – Fluxo das embarcações – cenário #5

Emb.	Vel. [nós]	Ori- gem	Des- tino	Dist. [milha]	Tempo Viagem [dia]	Custo Variável [\$]	Instante Chegada [dia]	Duração [dia]	LI [dia]	Início Tarefa [dia]	Instante Saída [dia]
1	12,0	0	5	220,80	0,77	1.533,33	1,77	17,00	1,00	1,77	18,77
1	12,0	5	7	0,00	0,00	0,00	18,77	10,00	15,00	19,21	29,21
1	12,0	7	11	220,80	0,77	1.533,33	29,98	0,00	0,00	33,98	33,98
2	12,0	0	5	220,80	0,77	1.533,33	1,77	17,00	1,00	1,77	18,77
2	12,0	5	7	0,00	0,00	0,00	18,77	10,00	15,00	19,21	29,21
2	12,0	7	11	220,80	0,77	1.533,33	29,98	0,00	0,00	33,98	33,98
3	12,0	0	9	292,26	1,01	2.029,58	2,01	3,00	1,00	2,01	5,01
3	12,0	9	8	221,37	0,77	1.537,29	5,78	20,00	1,00	5,78	25,78
3	12,0	8	11	324,83	1,13	2.255,76	26,91	0,00	0,00	33,98	33,98
4	12,0	0	9	292,26	1,01	2.029,58	2,01	3,00	1,00	2,01	5,01
4	12,0	9	1	59,91	0,21	416,04	5,22	11,00	12,00	17,28	28,28
4	12,0	1	11	269,22	0,93	1.869,58	29,22	0,00	0,00	33,98	33,98
5	12,5	0	2	265,07	0,88	1.877,58	1,88	13,00	1,00	1,88	14,88
5	12,5	2	4	276,48	0,92	1.958,40	15,81	6,00	7,00	15,81	21,81
5	12,5	4	11	241,03	0,80	1.707,30	22,61	0,00	0,00	33,98	33,98
6	12,5	0	2	265,07	0,88	1.877,58	1,88	13,00	1,00	1,88	14,88
6	12,5	2	1	6,40	0,02	45,33	14,90	11,00	12,00	17,28	28,28
6	12,5	1	11	269,22	0,90	1.906,98	29,18	0,00	0,00	33,98	33,98
7	12,5	0	2	265,07	0,88	1.877,58	1,88	13,00	1,00	1,88	14,88
7	12,5	2	4	276,48	0,92	1.958,40	15,81	6,00	7,00	15,81	21,81
7	12,5	4	10	57,20	0,19	405,17	22,00	11,00	12,00	22,00	33,00
7	12,5	10	11	296,70	0,99	2.101,63	33,98	0,00	0,00	33,98	33,98
8	12,5	0	3	225,83	0,75	1.599,63	1,75	17,00	1,00	1,75	18,75
8	12,5	3	10	87,66	0,29	620,93	19,04	11,00	12,00	22,00	33,00
8	12,5	10	11	296,70	0,99	2.101,63	33,98	0,00	0,00	33,98	33,98
9	14,0	0	3	225,83	0,67	2.016,34	1,67	17,00	1,00	1,75	18,75
9	14,0	3	7	153,69	0,46	1.372,23	19,21	10,00	15,00	19,21	29,21
9	14,0	7	11	220,80	0,66	1.971,43	29,87	0,00	0,00	33,98	33,98
10	14,5	0	9	292,26	0,84	2.666,45	1,84	3,00	1,00	2,01	5,01
10	14,5	9	4	247,62	0,71	2.259,18	5,73	6,00	7,00	15,81	21,81
10	14,5	4	10	57,20	0,16	521,87	21,97	11,00	12,00	22,00	33,00
10	14,5	10	11	296,70	0,85	2.706,96	33,85	0,00	0,00	33,98	33,98
11	14,7	0	6	184,20	0,52	1.694,24	1,52	13,00	4,00	4,00	17,00
11	14,7	6	1	100,32	0,28	922,73	17,28	11,00	12,00	17,28	28,28
11	14,7	1	11	269,22	0,76	2.476,24	29,05	0,00	0,00	33,98	33,98
12	14,7	0	6	184,20	0,52	1.694,24	1,52	13,00	4,00	4,00	17,00
12	14,7	6	1	100,32	0,28	922,73	17,28	11,00	12,00	17,28	28,28
12	14,7	1	11	269,22	0,76	2.476,24	29,05	0,00	0,00	33,98	33,98

Tabela 5.12 – Nível de serviço da solução – cenário #5

Tarefa	Instante Início [dia]	Instante Desejado [dia]	Atraso [dia]	Penalidade [\$/dia]	Atraso x Penal [\$]
1	17,28	17,00	0,28	118.760,00	33.769,85
2	1,88	3,00	0,00	303.050,00	0,00
3	1,75	3,00	0,00	94.459,00	0,00
4	15,81	12,00	3,81	227.398,00	865.287,29
5	1,77	6,00	0,00	7.078,00	0,00
6	4,00	9,00	0,00	192.547,00	0,00
7	19,21	20,00	0,00	242.608,00	0,00
8	5,78	6,00	0,00	93.031,00	0,00
9	2,01	4,00	0,00	199.945,00	0,00
10	22,00	17,00	5,00	4.652,00	23.240,62

O cálculo do custo total da solução será dado pela soma das parcelas de custo variável (tabela 5.11) e a soma das parcelas do atraso ponderado pela penalidade (tabela 5.12). Assim, o custo total = R\$ 60.010,18 + R\$ 922.297,76 = R\$ 982.307,94 (0,2% a mais que o valor indicado na tabela 5.5, por questões de arredondamento).

A solução pode ser visualizada nas figuras 5.2 e 5.3 por meio de um gráfico de Gantt; na figura 5.2, apenas as tarefas são representadas; na figura 5.3, foi incluído o fluxo de embarcações. Nestes gráficos, cada tarefa está representada por uma linha horizontal, cuja escala é o tempo. Os limites inferior e superior da janela de tempo estão identificados pelo par de colchetes. O instante desejado de início é o centro da janela de tempo, marcado por um traço vertical. A barra azul está posicionada horizontalmente no instante de tempo em que cada tarefa tem início, e o comprimento da barra é proporcional à sua duração. O gráfico da figura 5.3 contém, também, o fluxo das embarcações. Por serem 12 embarcações, a visualização fica comprometida, uma vez que viagens com mesma origem e mesmo destino, com embarcações de mesma velocidade, ficam sobrepostas. A figura 5.4 contém o gráfico de Gantt das embarcações. Cada linha representa uma embarcação e contém dois tipos de atividades: uma indicativa da viagem (barra escura) e outra representativa da tarefa (barra azul). Em cima da barra azul, consta o número da tarefa.

Figura 5.2 – Gráfico de Gantt das tarefas – cenário #5

Figura 5.3-- Gráfico de Gantt das tarefas com fluxo de embarcações – cenário #5

Figura 5.4— Gráfico de Gantt das embarcações contendo a sequência de tarefas realizadas – cenário #5

5.4 Heurística Construtiva – Versão Básica (HC Básica)

Os problemas também foram resolvidos por duas versões de uma heurística construtiva, descrita no capítulo 4. Nesta seção serão apresentados os resultados para a versão básica e na seção seguinte, os resultados da versão gulosa.

Os testes com as heurísticas construtivas (ambas as versões) têm por objetivo a construção de soluções viáveis para o conjunto de problemas, já que o método “branch and cut”, aplicado tanto para o modelo de fluxo em rede completo, como na versão compacta, foi incapaz de gerar pelo menos uma solução viável para, 14 e 11 problemas, respectivamente.

Assim, espera-se que o uso da estratégia heurística, que foi desenvolvida explorando as características principais do problema, redunde em um maior número de cenários resolvidos – isto é, gere ao menos uma solução viável para um maior número de problemas.

A eventual solução que for obtida pela aplicação da heurística construtiva poderá ainda ser objeto de melhoria. Pretende-se, portanto, aplicar posteriormente à solução heurística algum método que permita o seu refinamento. Três mecanismos serão empregados, a saber, os métodos “branch and cut”, “variable neighborhood search” e “local branching” (ver seções 5.6, 5.7 e 5.8). Desta forma, os tempos de processamento das heurísticas construtivas serão de 6 horas, assim como os dos métodos de refinamento. Isto permitirá a comparação destes métodos “compostos” com o método “branch and cut” isoladamente, para o qual o tempo de processamento foi limitado a 12 horas.

Os processamentos empregando a heurística construtiva – versão básica, serão conduzidos utilizando a sistemática apresentada no capítulo 4, em que o algoritmo é

executado diversas vezes, variando-se a semente do gerador de números aleatórios e o valor dos parâmetros usados nas distribuições. Além do tempo limite de processamento de 6 horas, poderá haver interrupção do processamento se o valor da função objetivo ficar a uma distância inferior a 1% de um limitante conhecido – podendo este ser o valor fornecido pelo pacote computacional CPLEX³⁵.

A tabela 5.13 lista os diversos valores adotados para cada parâmetro do algoritmo. Recapitulando, cada parâmetro tem a seguinte função na estrutura da heurística:

- *lim1* – indica o número máximo de tarefas sorteadas, quando estas estão ordenadas por instante de chegada;
- *lim2* – é um fator de segurança para cálculo da ‘tolerância negativa’;
- *lim3* – indica o número máximo de tarefas sorteadas, quando estas estão ordenadas segundo o critério de ‘folga’ ou pelo critério do ‘priorizador’;
- *lim4* – indica o limite para sorteio do número de vezes que a escolha de um conjunto de embarcações será refinada;
- *lim5* – indica o limite para sorteio do parâmetro de segurança ‘seg’ empregado no cálculo da folga de uma tarefa;
- *p1* – probabilidade de referência para escolha entre os critérios ‘menor tolerância’ ou ‘qualquer tolerância negativa’;
- *p2* – probabilidade de referência para escolha entre os critérios ‘menor folga’ ou ‘menor priorizador’.

³⁵ O intuito de usar os limitantes fornecidos pelo CPLEX visando interromper o processamento da heurística, quando o valor da função objetivo ficar a uma distância inferior a 1% do limitante, é para poupar o uso de recurso computacional.

Tabela 5.13 – Parâmetros testados na heurística construtiva – versão básica

<i>lim1</i>	8	4	3		
<i>lim2</i>	80%	100%	120%		
<i>lim3</i>	8	4	3	2	
<i>lim4</i>	3	5	7		
<i>lim5</i>	6	3			
<i>p1</i>	99%	0%	75%	50%	25%
<i>p2</i>	99%	0%	75%	50%	25%

Cada combinação dos valores destes parâmetros gera uma configuração específica, que será testada por um determinado número de iterações. O número de configurações geradas, portanto, será de 5400 ($= 3 \times 3 \times 4 \times 3 \times 2 \times 5 \times 5$). O número de vezes que a heurística é processada segundo uma determinada configuração foi fixado em 10. Este valor foi determinado de forma a permitir que os cenários contendo 40 tarefas pudessem ter as 5400 configurações testadas dentro do limite de tempo de 6 horas. Para os cenários contendo um menor número de tarefas, cujo tempo de processamento de cada solução heurística é menor, após as 5400 configurações terem sido testadas, inicia-se este processo novamente, ou seja, as configurações são novamente repetidas, até que o critério de parada seja atingido. Como a semente usada na geração dos números aleatórios é atualizada ao longo do processamento da heurística, a repetição de uma configuração com uma semente nova poderá gerar resultados diferentes.

O caráter probabilístico desta heurística requer que múltiplos processamentos sejam realizados. Assim, cada problema será executado 5 vezes, com o intuito de que soluções diferentes sejam obtidas.

A tabela 5.14 registra os valores da função objetivo das soluções heurísticas para cada processamento, apresentando também a média e o coeficiente de variação dos resultados obtidos.

Tabela 5.14 – Resultados da heurística construtiva – versão básica

Cenário	Teste 1 [\$]	Teste 2 [\$]	Teste 3 [\$]	Teste 4 [\$]	Teste 5 [\$]	Média [\$]	Coef. Variação
1	222.490,98	222.232,11	222.669,82	223.057,54	222.372,61	222.564,61	0,1%
2	322.389,67	321.915,66	322.689,22	322.069,12	322.381,30	322.288,99	0,1%
3	124.116,36	127.228,79	132.789,13	138.577,53	141.126,93	132.767,75	5,4%
4	306.114,61	306.052,21	306.440,30	307.616,41	307.311,68	306.707,04	0,2%
5	989.113,58	984.922,71	989.050,86	989.328,55	989.307,32	988.344,60	0,2%
6	185.772,35	186.194,62	186.569,90	186.175,05	186.486,54	186.239,69	0,2%
7	401.329,05	403.750,21	406.012,33	404.072,87	405.064,95	404.045,88	0,4%
8	731.415,31	732.595,23	731.154,33	731.447,67	731.446,01	731.611,71	0,1%
9	239.681,21	238.850,37	240.634,14	238.584,22	240.468,54	239.643,70	0,4%
10	210.159,01	210.441,32	210.441,32	210.462,53	210.528,19	210.406,47	0,1%
11	1.015.392,96	1.016.693,10	1.015.030,86	1.015.210,01	1.012.864,75	1.015.038,34	0,1%
12	306.563,31	294.960,63	301.370,74	277.566,97	288.310,77	293.754,48	3,9%
13	1.365.969,76	1.501.651,73	1.531.011,86	1.347.338,40	1.369.689,10	1.423.132,17	6,1%
14	856.509,91	893.242,14	943.726,21	825.374,64	776.064,40	858.983,46	7,4%
15	604.531,97	611.187,20	379.517,49	328.117,70	696.529,44	523.976,76	30,6%
16	2.157.691,68	1.972.111,00	2.118.015,14	2.117.705,74	2.081.265,99	2.089.357,91	3,4%
17	1.582.918,75	1.974.873,65	1.595.075,12	1.581.083,43	1.726.498,72	1.692.089,93	10,0%
18	1.445.178,17	1.625.894,31	1.453.397,44	1.591.490,49	1.364.937,21	1.496.179,52	7,3%
19	684.423,14	614.186,52	705.498,50	716.662,70	615.187,77	667.191,73	7,4%
20	847.189,29	656.967,43	465.091,18	589.397,57	740.685,98	659.866,29	22,0%
21	778.835,32	666.930,52	910.908,75	629.584,66	568.152,02	710.882,25	19,1%
22	799.548,26	464.351,96	675.120,78	631.181,70	862.607,30	686.562,00	22,6%
23	920.869,06	884.663,09	856.885,89	688.242,62	933.759,07	856.883,95	11,6%
24	913.684,27	930.879,56	743.492,91	778.607,70	876.282,72	848.589,43	9,8%
25	1.483.859,61	1.091.615,51	1.469.375,83	1.092.830,39	1.452.705,87	1.318.077,44	15,7%
26	2.905.621,23	3.155.133,95	2.703.070,59	2.849.148,81	2.258.509,72	2.774.296,86	11,9%
27	505.675,13	261.781,53	366.780,15	381.541,33	529.236,45	409.002,92	26,8%
28	882.134,10	676.732,43	729.058,95	536.491,34	789.652,04	722.813,77	17,9%
29	706.776,12	591.818,46	342.177,36	608.876,29	419.879,43	533.905,53	27,9%
30	267.673,04	455.459,35	474.499,27	179.769,96	412.541,30	357.988,58	35,9%
31	1.986.481,41	1.779.746,49	1.968.968,63	1.519.754,15	2.075.316,01	1.866.053,34	11,9%
32	3.242.852,79	3.941.420,85	3.646.293,19	3.530.230,94	4.432.201,99	3.758.599,95	12,0%
33	2.976.259,76	4.631.598,99	4.750.334,54	4.153.377,86	3.328.168,00	3.967.947,83	19,8%
34	1.844.042,86	1.938.483,91	1.555.481,93	2.266.765,34	1.068.807,63	1.734.716,33	26,0%
35	6.831.405,67	7.253.355,63	7.531.102,87	6.813.747,94	5.430.102,71	6.771.942,96	11,9%
36	4.130.840,56	3.783.876,62	5.067.776,86	3.448.200,26	4.240.783,93	4.134.295,65	14,7%
37	4.848.481,82	4.249.145,33	5.488.280,74	4.056.352,95	4.946.955,96	4.717.843,36	12,2%
38	2.357.360,35	1.723.284,75	1.935.251,84	1.421.661,02	2.007.594,43	1.889.030,48	18,4%
39	1.386.749,23	909.450,35	1.113.194,48	1.150.287,05	1.617.004,77	1.235.337,18	22,1%
40	4.833.548,46	3.753.034,33	4.537.760,00	4.090.829,95	3.994.637,83	4.241.962,11	10,3%
41	5.481.738,36	4.298.873,38	3.767.379,75	3.190.959,85	4.327.293,52	4.213.248,97	20,1%
42	536.135,59	803.916,66	538.593,60	991.176,48	486.565,66	671.277,60	32,5%
43	1.791.045,51	1.778.672,72	990.130,59	1.447.929,66	2.452.702,17	1.692.096,13	31,7%
44	2.183.065,10	1.892.115,80	1.858.519,96	2.159.316,27	2.141.247,32	2.046.852,89	7,7%
45	585.961,92	980.137,14	1.333.042,59	1.017.854,76	1.093.136,60	1.002.026,60	27,0%
46	3.320.695,35	1.999.227,87	3.326.625,56	2.789.714,52	3.040.685,85	2.895.389,83	18,9%
47	5.156.530,18	5.877.270,43	5.944.951,51	6.052.246,68	3.896.796,23	5.385.559,01	16,8%
48	3.143.457,31	2.710.217,52	2.665.528,05	2.904.023,68	3.506.485,66	2.985.942,44	11,6%

Observa-se na tabela 5.14 que a heurística conseguiu encontrar uma solução viável para todos os cenários; a melhor solução para cada cenário é indicada em negrito. A dispersão dos resultados, medido pelo coeficiente de variação (desvio padrão / média) indica um valor médio de 13,1%, considerado significativo. Este fato mostra que, outros processamentos poderão, eventualmente, gerar soluções ainda melhores, em função da variabilidade inerente à heurística.

O cálculo do “gap” de cada solução foi efetuado, tendo como referência o limitante obtido pelo método “branch and cut”. O valor médio do “gap”, para todos os cenários, foi 62,3%, prejudicado, em parte, pelas estimativas pobres fornecidas pelo pacote computacional CPLEX para os problemas de maior porte (30 e 40 tarefas). Para os problemas de 10 tarefas, o valor médio do “gap” foi 6,7%, lembrando que o resultado do cenário 10 tinha apresentado um “gap” de 46,9% e 46,8% em relação a seu limitante inferior, após o processamento do método “branch and cut” para versões completa e compacta do modelo de fluxo em rede, respectivamente. Se o cenário 10 for desconsiderado, o valor médio do “gap” do conjunto de 10 tarefas cai para 3,0%, no caso da versão básica da heurística construtiva.

Para que os resultados da heurística construtiva – versão básica possam ser comparados aos resultados obtidos pelo método “branch and cut” (melhor solução entre os modelos completo e compacto), é necessário que nove cenários, para os quais não se obteve solução por este método, sejam excluídos. A heurística apresentou um valor médio da função objetivo de 13,6% a mais que os valores obtidos pelo “branch and cut”. Segmentando esta análise por classe de problemas de 10, 20, 30 e 40 tarefas, a heurística gerou soluções com valores médios de, respectivamente, +2,4%, +36,1%, +23,6% e -23,9% em relação ao “branch and cut”.

A figura 5.5 exemplifica a evolução da função objetivo para o cenário 5. Neste problema, o tempo total de processamento foi da ordem de 950 segundos, e a última solução gerada ficou a uma distância inferior a 1% do limitante informado.

Figura 5.5 – Evolução da heurística construtiva – versão básica, aplicada ao cenário

#5

Os parâmetros utilizados na heurística construtiva que levaram aos melhores resultados, por cenário, estão destacados na tabela 5.15. A tabela 5.16 agrupa estes valores, mostrando o número de ocorrências de cada parâmetro na solução final, permitindo inferir a faixa de valores em que a heurística funciona melhor.

Tabela 5.15 – Parâmetros da heurística construtiva – versão básica, da melhor solução de cada cenário

Cenário	<i>lim1</i>	<i>lim2</i>	<i>lim3</i>	<i>lim4</i>	<i>lim5</i>	<i>p1</i>	<i>p2</i>
1	4	80%	2	7	6	99%	75%
2	4	100%	2	7	6	99%	75%
3	4	120%	2	7	6	99%	0%
4	4	80%	2	7	6	99%	0%
5	8	80%	3	5	6	99%	99%
6	8	100%	4	5	3	99%	75%
7	4	120%	2	7	6	0%	99%
8	8	100%	4	3	6	75%	50%
9	8	120%	3	7	6	0%	0%
10	8	120%	2	5	3	99%	50%
11	8	80%	3	3	6	99%	75%
12	8	100%	2	7	6	0%	99%
13	8	100%	2	7	6	99%	75%
14	8	100%	2	7	3	50%	50%
15	4	100%	2	7	3	50%	0%
16	4	100%	3	7	3	0%	99%
17	8	120%	2	5	3	0%	50%
18	8	120%	4	5	3	99%	99%
19	8	100%	2	7	3	25%	0%
20	4	120%	2	7	6	0%	0%
21	8	120%	2	5	3	0%	75%
22	8	120%	4	7	6	99%	50%
23	4	100%	4	7	6	50%	75%
24	4	120%	2	7	6	0%	50%
25	8	120%	4	7	3	99%	25%
26	8	80%	2	7	6	0%	0%
27	4	100%	2	7	6	75%	0%
28	4	80%	2	7	3	99%	50%
29	4	80%	2	7	3	0%	75%
30	4	80%	2	7	3	99%	99%
31	8	80%	2	7	6	50%	99%
32	4	100%	2	7	6	99%	0%
33	8	100%	3	7	6	99%	75%
34	8	120%	2	7	6	50%	99%
35	4	100%	3	7	3	25%	99%
36	4	80%	2	7	6	25%	99%
37	8	120%	2	7	3	50%	99%
38	4	100%	2	7	3	25%	50%
39	4	120%	2	7	3	0%	0%
40	8	80%	2	7	6	50%	25%
41	4	100%	3	7	6	0%	75%
42	8	80%	2	7	3	99%	50%
43	8	80%	2	7	3	75%	0%
44	4	80%	2	7	3	75%	75%
45	8	80%	2	7	6	50%	99%
46	8	100%	3	7	3	75%	75%
47	8	100%	2	7	6	25%	99%
48	8	100%	2	7	6	25%	50%

Tabela 5.16 – Parâmetros usados nas soluções de menor custo

<i>lim1</i>	#	<i>lim2</i>	#	<i>lim3</i>	#	<i>lim4</i>	#	<i>lim5</i>	#	<i>p1</i>	#	<i>p2</i>	#
3	0	80%	15	2	34	3	2	3	21	0%	12	0%	11
4	21	100%	19	3	8	5	6	6	27	25%	6	25%	2
8	27	120%	14	4	6	7	40			50%	8	50%	10
										75%	5	75%	12
										99%	17	99%	13

- Número de ocorrências

O parâmetro *lim1* indica o valor máximo para sorteio quando as tarefas estão ordenadas por data, ou seja, o sorteio é feito escolhendo uma dentre as *lim1* primeiras tarefas em fila. Em 27 casos (56,3%), a melhor solução foi encontrada com *lim1*=8, mostrando que, não necessariamente, as tarefas cujo instante de início é menor, deverão ter maior prioridade.

O parâmetro *lim3* indica o valor máximo para sorteio quando as tarefas estão ordenadas segundo o critério de ‘folga’ ou pelo critério do ‘priorizador’. Nestes casos, a forma mais eficiente de escolha da tarefa consistiu em sortear entre as *duas* tarefas com menor ‘folga’ ou menor ‘priorizador’, isto é, as tarefas mais críticas da fila com relação ao aspecto da janela de tempo.

O parâmetro *lim4* indica o valor máximo usado no sorteio do número de vezes que a escolha do conjunto de embarcações será refinada. As melhores soluções obtidas foram com um elevado número de refinamentos (40 ocorrências), permitindo concluir que a aplicação de um critério guloso na escolha do conjunto poderia levar a soluções de boa qualidade, já que aplicando o refinamento várias vezes, a chance escolher o melhor conjunto (isto é, o conjunto que inicia a tarefa no instante mais cedo) é alta. Portanto, existe uma semelhança em aplicar um critério guloso ou refinar a escolha de um conjunto várias vezes.

A análise dos valores testados para os demais parâmetros (*lim2*, *lim5*, *p1* e *p2*) não indicou uma tendência específica.

5.5 Heurística Construtiva – Versão Gulosa (HC Gulosa)

A versão gulosa da heurística construtiva difere da versão básica no critério de escolha do conjunto de embarcações. Ao invés de sortear um conjunto viável e aplicar a rotina de refinamento da escolha, conforme descrito no capítulo 4, é introduzida uma probabilidade de se escolher o melhor conjunto de embarcações, dentre os viáveis, isto é, aquele em que as embarcações se reposicionam mais rapidamente.

A aplicação da regra gulosa a todas as tarefas da fila cria um processo de escolha “míope”, pois a ‘primeira’ tarefa escolhida da fila levaria vantagem em alocar as embarcações mais rápidas ou mais próximas a ela. Caso haja uma tarefa mais importante (em função da penalidade) ou mais crítica (quanto à janela de tempo) na fila, esta poderia ser prejudicada pela falta do recurso embarcação. Desta forma, o emprego da estratégia gulosa para a escolha do conjunto de embarcações ocorrerá com probabilidade p_3 , sendo p_3 um parâmetro de entrada. Ter-se-á, assim, uma heurística míope probabilística, ou parcialmente míope.

Os demais parâmetros empregados na heurística são os mesmos indicados na seção anterior, na “versão básica”. A única modificação foi quanto à faixa de valores testados para as probabilidades de referência p_1 e p_2 , que foram reduzidas para não criar um número total de configurações elevado. A lista dos valores adotados para os parâmetros empregados na implementação da versão gulosa está na tabela 5.17, implicando um total de 5832 ($= 3 \times 3 \times 4 \times 3 \times 2 \times 3 \times 3 \times 3$) configurações. Cada configuração também será testada durante 10 iterações.

Tabela 5.17 - Parâmetros testados na heurística construtiva – versão gulosa

<i>lim1</i>	8	4	3	
<i>lim2</i>	80%	100%	120%	
<i>lim3</i>	8	4	3	2
<i>lim4</i>	3	5	7	
<i>lim5</i>	6	3		
<i>p1</i>	75%	50%	25%	
<i>p2</i>	75%	50%	25%	
<i>p3</i>	75%	50%	25%	

A tabela 5.18 lista, para cada cenário, a função objetivo dos cinco processamentos, destacando em negrito a solução de mínimo custo de cada cenário. Novamente, foi possível obter pelo menos uma solução para todos os problemas testados. A dispersão observada foi similar à da heurística construtiva – versão básica, sendo o valor médio do coeficiente de variação igual a 13,3%.

Assim como na versão básica, o cálculo do “gap” de cada solução foi efetuado, tendo como referência o limitante obtido pelo método “branch and cut”. O valor médio do “gap”, para todos os cenários, foi 63,3%, apresentando variação de +0,1% em relação à versão básica. Para os problemas de 10 tarefas, o valor médio do “gap” foi 5,8%, lembrando que o resultado do cenário 10 tinha apresentado um “gap” de 46,9% e 46,8% em relação a seu limitante inferior, após o processamento do método “branch and cut” para versões completa e compacta do modelo de fluxo em rede, respectivamente. Se o cenário 10 for desconsiderado, o valor médio do “gap” do conjunto de 10 tarefas cai para 2,0%, no caso da versão básica da heurística gulosa.

A versão gulosa da heurística construtiva foi melhor que a versão básica em 26 dos 48 problemas. Contudo, em média, os valores da função objetivo da versão básica são 1,6% menores.

Tabela 5.18 - Resultados da heurística construtiva – versão gulosa

Cená- rio	Teste 1 [\$]	Teste 2 [\$]	Teste 3 [\$]	Teste 4 [\$]	Teste 5 [\$]	Média [\$]	Coef. Variação
1	222.600,66	222.500,82	222.648,79	223.677,60	222.986,34	222.882,84	0,2%
2	322.593,60	321.852,86	322.884,21	321.637,70	323.760,09	322.545,69	0,3%
3	137.092,67	140.915,14	133.598,97	135.489,51	122.852,10	133.989,68	5,1%
4	306.386,17	307.710,09	306.119,74	307.737,37	307.105,59	307.011,79	0,2%
5	984.763,82	988.652,52	988.795,31	989.362,43	984.889,63	987.292,74	0,2%
6	186.843,90	187.355,40	187.140,52	186.075,75	186.472,90	186.777,69	0,3%
7	405.889,12	401.898,86	401.971,58	401.902,45	404.171,73	403.166,75	0,4%
8	634.070,89	731.461,06	732.407,20	731.722,54	731.433,00	712.218,94	6,1%
9	240.970,43	238.543,15	241.653,14	239.771,39	239.182,32	240.024,09	0,5%
10	210.396,62	210.378,27	210.978,75	210.193,66	210.148,98	210.419,26	0,2%
11	1.012.640,96	1.015.962,95	1.016.783,06	1.014.085,56	1.013.921,80	1.014.678,87	0,2%
12	286.419,57	309.756,88	293.065,35	294.645,67	305.368,59	297.851,21	3,2%
13	1.483.784,47	1.334.132,37	1.587.355,73	1.323.858,67	1.168.493,09	1.379.524,87	11,7%
14	907.421,60	900.985,08	936.238,44	919.099,62	802.843,52	893.317,65	5,9%
15	646.863,71	506.633,50	221.739,00	703.276,49	525.853,57	520.873,25	35,8%
16	2.037.475,61	1.966.349,49	2.078.708,17	2.171.232,58	2.009.742,53	2.052.701,68	3,8%
17	1.907.210,45	1.669.238,09	1.858.401,63	1.342.055,80	1.560.460,83	1.667.473,36	13,8%
18	1.487.223,85	1.559.115,66	1.480.221,24	1.569.695,27	1.450.118,93	1.509.274,99	3,5%
19	665.436,79	692.538,96	688.755,08	673.074,15	654.700,47	674.901,09	2,3%
20	596.502,17	596.277,20	914.855,50	505.794,93	616.993,77	646.084,71	24,2%
21	544.966,86	850.700,44	749.127,92	637.535,84	900.359,44	736.538,10	20,0%
22	421.103,40	600.620,73	865.078,51	494.537,21	602.918,62	596.851,69	28,2%
23	901.361,27	668.374,14	860.128,84	653.564,80	997.036,98	816.093,21	18,4%
24	829.198,13	814.668,88	1.085.065,38	981.250,59	911.770,14	924.390,62	12,1%
25	1.187.767,16	894.144,48	1.249.816,94	1.594.039,64	1.307.034,77	1.246.560,60	20,1%
26	2.470.306,54	2.531.756,97	2.101.298,45	3.106.158,13	3.040.434,00	2.649.990,82	15,9%
27	343.139,05	460.317,72	314.518,48	438.874,77	433.521,06	398.074,22	16,3%
28	625.207,65	770.188,37	427.524,64	902.044,47	565.451,44	658.083,31	27,9%
29	578.527,06	497.444,22	525.896,60	519.030,12	445.788,38	513.337,28	9,4%
30	374.692,08	539.946,85	345.171,37	382.485,04	446.479,31	417.754,93	18,6%
31	1.401.021,93	1.275.137,61	1.732.420,76	1.587.844,98	2.067.923,96	1.612.869,85	19,1%
32	3.170.537,77	3.377.604,00	4.172.977,29	3.427.654,83	3.289.028,17	3.487.560,41	11,3%
33	4.558.372,40	3.375.938,17	4.048.446,49	4.165.495,99	3.584.275,15	3.946.505,64	11,9%
34	1.979.817,61	2.471.724,21	1.255.833,88	1.635.649,46	1.977.323,93	1.864.069,82	24,3%
35	6.663.359,44	6.972.490,95	7.298.386,93	7.996.631,19	6.180.223,98	7.022.218,50	9,7%
36	4.253.551,41	5.211.407,59	4.060.579,14	4.237.180,32	4.659.887,31	4.484.521,15	10,3%
37	5.436.714,52	4.410.601,29	4.102.716,54	4.712.244,32	5.086.414,70	4.749.738,27	11,1%
38	2.584.634,70	1.810.065,90	1.277.541,12	1.712.476,14	2.698.534,29	2.016.650,43	30,1%
39	1.091.939,58	784.203,96	1.188.498,64	1.365.706,23	1.315.563,72	1.149.182,43	20,1%
40	2.704.305,47	3.533.501,45	4.282.897,31	3.907.235,28	3.957.048,30	3.676.997,56	16,5%
41	3.706.154,91	4.237.392,74	5.086.306,45	5.805.063,02	4.874.760,34	4.741.935,49	17,0%
42	604.764,41	918.036,32	575.927,65	959.266,08	851.613,22	781.921,54	22,9%
43	2.485.615,71	1.857.902,90	1.885.330,17	1.555.569,35	1.330.596,19	1.823.002,86	23,9%
44	2.290.865,33	2.381.271,12	1.209.300,50	2.384.393,03	1.766.732,23	2.006.512,44	25,6%
45	716.554,90	492.293,54	1.268.817,02	781.565,11	765.346,56	804.915,43	35,3%
46	3.397.815,14	3.602.436,12	2.942.312,64	3.035.151,71	3.104.148,24	3.216.372,77	8,6%
47	4.945.184,99	3.702.733,21	5.699.290,60	4.781.529,18	4.448.270,11	4.715.401,62	15,4%
48	2.952.498,13	2.265.180,53	2.167.414,40	3.399.714,86	2.215.649,29	2.600.091,44	21,2%

Para que os resultados da heurística construtiva – versão gulosa possam ser comparados aos resultados obtidos pelo método “branch and cut” (melhor solução entre os modelos completo e compacto), é necessário que nove cenários, para os quais não se obteve solução por este método, sejam excluídos. A heurística apresentou um valor médio da função objetivo de 13,2% a mais que os valores obtidos pelo “branch and cut”. Segmentando esta análise por classe de problemas de 10, 20, 30 e 40 tarefas, a heurística gerou soluções com valores médios de, respectivamente, +1,4%, +23,1%, +43,6% e -28,6% em relação ao “branch and cut”.

A figura 5.6 exemplifica a evolução da função objetivo para o cenário 14. Neste caso, o critério de parada da heurística foi por tempo máximo (6 horas).

Figura 5.6 - Evolução da heurística construtiva – versão gulosa, aplicada ao cenário

Os parâmetros utilizados na versão gulosa da heurística construtiva que levaram aos melhores resultados, em cada cenário, estão destacados na tabela 5.19. A tabela 5.20 agrupa estes valores, mostrando o número de ocorrências de cada parâmetro na solução final, permitindo inferir a faixa de valores em que a heurística funciona melhor.

Os parâmetros que influenciam a escolha do conjunto de embarcações, $p3$ e $lim4$, indicam fatos importantes. Como um maior valor de $p3$ redundava em maior frequência da aplicação da regra gulosa, observa-se que em apenas 12 problemas a melhor solução foi alcançada com uma baixa utilização da regra gulosa ($p3=25\%$). Mesmo assim, independente da aplicação da regra gulosa, o parâmetro que indica o número máximo de vezes que uma solução é refinada ($lim4$) aparece 39 vezes com valor 7, mostrando que as melhores soluções obtidas requerem um número maior de refinamentos.

Por último, considerando como solução heurística a melhor solução obtida entre as versões básica e gulosa, o valor médio do “gap” heurístico é de 61,8% e, os valores da função objetivo ficaram, em média, 5,4% acima dos melhores resultados do método “branch and cut”.

5.6 Heurística Construtiva & “Branch and Cut”

Com o objetivo de avaliar o desempenho de estratégias “compostas” ou “híbridas”, aplicou-se o método “branch and cut” à melhor solução heurística obtida. Para permitir uma comparação coerente com o método “branch and cut”, os tempos de processamento da heurística e do método de melhoria foram ambos iguais a 6 horas.

Tabela 5.19 – Parâmetros da heurística construtiva – versão gulosa, da melhor solução de cada cenário

Cenário	<i>lim1</i>	<i>lim2</i>	<i>lim3</i>	<i>lim4</i>	<i>lim5</i>	<i>p1</i>	<i>p2</i>	<i>p3</i>
1	4	80%	2	7	3	75%	75%	75%
2	4	80%	3	7	6	75%	50%	25%
3	4	80%	2	5	3	75%	75%	50%
4	8	120%	2	3	6	25%	75%	75%
5	8	120%	2	7	3	75%	75%	25%
6	3	80%	4	5	6	50%	25%	75%
7	4	80%	4	7	6	75%	75%	50%
8	4	80%	4	7	6	75%	50%	75%
9	4	80%	2	7	6	75%	75%	50%
10	4	120%	3	5	3	75%	50%	75%
11	8	80%	2	5	6	75%	25%	50%
12	3	120%	2	7	6	50%	75%	25%
13	8	100%	3	7	6	25%	50%	75%
14	4	80%	2	7	3	25%	50%	25%
15	4	100%	3	7	3	50%	50%	75%
16	3	100%	3	7	3	75%	25%	75%
17	3	80%	4	7	6	25%	75%	50%
18	3	120%	3	5	6	75%	75%	75%
19	3	100%	3	7	3	25%	75%	50%
20	8	120%	3	7	6	25%	75%	50%
21	8	80%	4	7	3	50%	75%	50%
22	3	120%	3	7	6	75%	50%	50%
23	3	80%	2	5	6	25%	75%	25%
24	8	120%	2	5	6	50%	25%	75%
25	4	80%	2	7	6	50%	75%	75%
26	4	80%	3	7	3	75%	75%	25%
27	8	80%	4	7	6	75%	25%	25%
28	8	80%	4	7	6	50%	25%	25%
29	4	100%	2	7	6	25%	75%	25%
30	4	120%	3	7	3	25%	75%	50%
31	8	100%	3	7	6	50%	50%	25%
32	4	80%	4	7	6	75%	50%	50%
33	3	100%	3	7	3	50%	75%	25%
34	3	100%	4	7	3	75%	75%	50%
35	8	120%	4	7	3	25%	50%	75%
36	8	100%	2	7	6	75%	25%	50%
37	4	80%	2	7	6	25%	50%	75%
38	4	80%	2	7	3	75%	75%	75%
39	4	100%	2	7	6	50%	50%	75%
40	4	100%	3	7	3	25%	50%	50%
41	4	120%	2	7	6	75%	25%	50%
42	8	80%	2	7	3	50%	75%	75%
43	3	100%	2	7	6	75%	50%	50%
44	4	120%	3	7	3	25%	25%	50%
45	8	80%	2	7	3	25%	25%	50%
46	8	100%	2	7	6	25%	25%	75%
47	4	80%	2	5	3	25%	50%	25%
48	8	100%	2	7	3	75%	25%	50%

Tabela 5.20 – Parâmetros usados nas soluções de menor custo

<i>lim1</i>	#	<i>lim2</i>	#	<i>lim3</i>	#	<i>lim4</i>	#	<i>lim5</i>	#	<i>p1</i>	#	<i>p2</i>	#	<i>p3</i>	#
3	11	80%	22	2	23	3	1	3	21	25%	16	25%	12	25%	12
4	21	100%	14	3	15	5	8	6	27	50%	11	50%	15	50%	19
8	16	120%	12	4	10	7	39			75%	21	75%	21	75%	17

- Número de ocorrências

A aplicação do método “branch and cut” a partir da solução inicial heurística será feita usando como modelo matemático aquele que apresentou melhor desempenho quando processado sem o fornecimento de uma solução inicial, ou seja, o modelo de fluxo em rede com a formulação compacta.

Na tabela 5.21 são apresentados, para cada cenário, os valores da função objetivo da melhor solução do método “branch and cut” (melhor resultado entre a formulação completa e a formulação compacta do modelo de fluxo em rede); da melhor solução heurística (melhor resultado entre a versão básica e a versão gulosa da heurística), e da melhor solução resultante da aplicação do método “branch and cut” à melhor solução heurística. Nesta tabela também são mostrados, para cada cenário, o melhor limitante inferior, o número de soluções obtidas, o número de nós explorados da árvore de solução e o valor do “gap”.

Os valores apresentados na tabela 5.21 indicam que o valor médio do “gap” da estratégia combinada *heurística construtiva & “branch and cut”* foi 54,7%, contra 63,2% e 63,3% das heurísticas construtivas, versão básica e gulosa, respectivamente. O valor médio da função objetivo da melhor solução heurística construtiva tinha ficado 5,4% acima do valor médio da melhor solução obtida pelo método “branch and cut”; a estratégia combinada fez a média cair para -13,4%, lembrando que esta análise foi aplicada para os 39 problemas para os quais se obteve pelo menos uma solução inteira pelo método “branch and cut”.

Tabela 5.21 - Resultados do método “branch and cut” aplicado ao modelo de fluxo em rede – formulação compacta, com solução inicial heurística

Cenário	Melhor BC [\$]	Melhor HC [\$]	HC&BC [\$]	Bound [\$]	# Sol.	Nós	GAP
1	221.478,92	222.232,11	221.518,08	219.423,69	3	779	0,95%
2	320.903,43	321.637,70	321.559,99	320.012,59	2	7	0,48%
3	121.749,50	122.852,10	121.749,50	120.624,70	2	861	0,92%
4	304.837,97	306.052,21	304.807,98	302.305,30	3	52	0,82%
5	979.990,95	984.763,82	981.220,10	977.786,19	2	281	0,35%
6	182.930,96	185.772,35	183.097,48	182.375,34	4	137	0,39%
7	401.906,02	401.329,05	401.906,02	397.989,15	2	4.389	0,97%
8	629.357,48	634.070,89	629.357,48	623.207,22	4	7.500	0,98%
9	237.501,42	238.543,15	237.171,66	235.883,06	4	68	0,54%
10	207.867,45	210.148,98	207.886,55	110.569,31	8	269.483	46,81%
11	1.009.664,51	1.012.640,96	1.010.878,12	1.005.869,64	3	37	0,50%
12	261.365,23	277.566,97	261.365,23	258.981,11	5	6.186	0,91%
13	528.829,60	1.168.493,09	757.403,54	103.460,71	9	30.158	86,34%
14	909.638,55	776.064,40	679.682,80	104.600,50	10	27.042	84,61%
15	107.582,21	221.739,00	107.588,11	100.798,86	19	48.835	6,31%
16	1.827.501,26	1.966.349,49	1.871.660,69	836.356,58	11	48.003	55,31%
17	556.027,95	1.342.055,80	643.507,02	409.679,92	28	51.711	36,34%
18	742.715,71	1.364.937,21	1.146.254,86	116.147,01	10	26.612	89,87%
19	715.186,72	614.186,52	484.798,70	91.124,48	10	23.200	81,20%
20	1.468.270,15	465.091,18	104.039,81	79.218,05	9	24.881	23,86%
21	4.107.550,39	544.966,86	476.047,77	83.966,34	6	13.609	82,36%
22	1.361.556,70	421.103,40	156.436,33	92.179,24	15	27.311	41,08%
23	444.872,75	653.564,80	504.085,24	132.499,32	10	30.259	73,71%
24	807.196,86	743.492,91	709.262,12	112.644,03	9	17.317	84,12%
25	1.891.799,93	894.144,48	860.079,71	92.262,69	4	11.203	89,27%
26	1.660.976,06	2.101.298,45	1.658.751,52	1.585.734,27	9	12.436	4,40%
27	162.732,27	261.781,53	137.494,64	113.644,97	15	15.205	17,35%
28	272.703,36	427.524,64	350.545,94	133.215,49	6	16.846	62,00%
29	129.730,26	342.177,36	341.356,12	113.395,78	1	11.061	66,78%
30	142.818,37	179.769,96	140.242,16	138.425,94	14	20.216	1,30%
31	3.425.403,80	1.275.137,61	1.275.853,13	83.382,64	1	7.569	93,46%
32		3.170.537,77	3.170.719,32	79.391,98	1	4.359	97,50%
33		2.976.259,76	2.977.925,52	96.186,69	1	4.794	96,77%
34	2.007.944,88	1.068.807,63	1.066.269,82	106.465,54	1	7.867	90,02%
35	7.388.233,59	5.430.102,71	5.429.430,28	203.516,78	1	4.633	96,25%
36		3.448.200,26	3.452.382,83	115.302,92	1	4.526	96,66%
37	2.400.857,11	4.056.352,95	3.509.911,72	837.441,35	2	6.318	76,14%
38		1.277.541,12	1.277.190,45	119.359,60	2	5.643	90,65%
39	5.245.226,34	784.203,96	785.029,32	137.477,49	1	4.648	82,49%
40	2.572.777,40	2.704.305,47	1.701.359,01	344.488,74	8	7.310	79,75%
41		3.190.959,85	3.132.253,44	133.939,73	4	5.753	95,72%
42	598.289,07	486.565,66	268.552,51	124.181,63	9	6.324	53,76%
43	10.005.453,44	990.130,59	975.165,33	104.870,47	3	3.787	89,25%
44		1.209.300,50	1.202.736,65	115.798,42	3	4.151	90,37%
45	1.763.845,32	492.293,54	417.676,59	102.533,83	2	3.582	75,45%
46		1.999.227,87	1.961.319,92	118.747,22	2	3.836	93,95%
47		3.702.733,21	3.350.962,08	196.218,86	4	3.704	94,14%
48		2.167.414,40	2.160.298,14	124.117,00	2	2.509	94,25%

Já os valores médios da função objetivo desta estratégia combinada em relação aos valores das soluções heurísticas, para os 48 problemas, apresentaram uma variação de -13,0%, indicando uma melhoria expressiva. Na execução do método “branch and cut” foram geradas, em média, 5,9 soluções por cenário durante o período de 6 horas. A figura 5.7 exemplifica a melhoria da função objetivo para o cenário 17, em que 28 soluções foram geradas a partir da solução inicial informada.

Figura 5.7 – Evolução do método “branch and cut” aplicado à solução inicial heurística, para o cenário #17

5.7 Heurística Construtiva & “Variable Neighborhood Search”

A estratégia de busca em vizinhança “variable neighborhood search” será testada apenas no modelo de fluxo em rede – formulação compacta, pois foi com este modelo que foram obtidos os melhores resultados com a estratégia “branch and cut”.

Este método utilizará como solução inicial a melhor solução heurística³⁶, e terá tempo de processamento de 6 horas. Assim, os dois métodos combinados poderão ser comparados com o desempenho do algoritmo “branch and cut” isoladamente.

A estratégia de busca proposta por Hansen et al. (2006) requer a calibração de alguns parâmetros. Além da duração total, de 6 horas, dois parâmetros influenciam diretamente a exploração da vizinhança: *k_passo* – indica o passo de crescimento da vizinhança; *limite_tempo_nó* – indica o tempo total em que a vizinhança será explorada. Para o primeiro parâmetro, os autores propõem valores na faixa de $3 \leq k_passo \leq 10$; para o segundo parâmetro, é sugerida a faixa entre 180 e 1200 segundos.

As seguintes combinações de valores serão testadas para os parâmetros (*k_passo*; *limite_tempo_nó* [s]): (3; 300), (5; 300), (5; 600), (7; 420). Apesar do método VNS poder apresentar resultados ainda melhores explorando-se uma combinação maior destes parâmetros, apenas as 4 configurações foram testadas, por limitação de recursos computacionais. Cada configuração foi aplicada ao conjunto de 48 problemas, cujos resultados são mostrados na tabela 5.22; os resultados de mínimo custo de cada cenário estão destacados em negrito.

A análise destes resultados mostra que, nos problemas de 10 tarefas, as quatro configurações de vizinhança levaram ao mesmo resultado final. Para os demais problemas houve dez casos de empate (mais de uma estrutura de vizinhança levando ao melhor resultado).

³⁶ Caso o método “variable neighborhood search” seja processado sem uma solução inicial, o algoritmo “branch and cut” será executado até que uma solução inteira seja gerada. Só então a busca em vizinhança iniciará.

Tabela 5.22 - Resultados do método “variable neighborhood search” aplicado ao modelo de fluxo em rede – formulação compacta, com solução inicial heurística

Cenário	3_300	5_300	5_600	7_420
1	221.478,92	221.478,92	221.478,92	221.478,92
2	320.903,43	320.903,43	320.903,43	320.903,43
3	121.749,50	121.749,50	121.749,50	121.749,50
4	304.807,98	304.807,98	304.807,98	304.807,98
5	979.632,35	979.632,35	979.632,35	979.632,35
6	182.898,78	182.898,78	182.898,78	182.898,78
7	401.906,02	401.906,02	401.906,02	401.906,02
8	629.357,48	629.357,48	629.357,48	629.357,48
9	237.141,21	237.141,21	237.141,21	237.141,21
10	207.867,45	207.867,45	207.867,45	207.867,45
11	1.009.457,92	1.009.457,92	1.009.457,92	1.009.457,92
12	261.365,23	261.365,23	261.365,23	261.365,23
13	697.342,94	857.146,06	702.636,01	741.427,50
14	445.474,28	503.963,02	523.457,85	331.196,05
15	107.623,00	107.623,00	107.623,00	107.623,00
16	1.779.165,09	1.779.169,92	1.779.192,34	1.779.191,46
17	623.335,08	623.335,08	623.346,34	623.337,68
18	796.155,97	512.792,19	790.215,15	474.254,59
19	559.784,30	516.125,36	503.647,28	503.729,40
20	183.381,14	183.604,87	183.607,66	183.606,78
21	254.033,12	211.201,20	201.938,15	209.797,91
22	141.982,05	142.662,40	143.341,17	142.929,79
23	231.428,39	231.832,63	231.868,28	235.419,42
24	551.090,21	324.787,96	550.997,00	550.948,24
25	540.602,04	541.907,88	453.276,14	545.474,29
26	1.752.082,07	1.677.790,20	1.673.792,91	1.674.588,30
27	246.702,38	174.788,74	171.219,00	224.072,34
28	372.608,98	372.608,98	391.505,17	409.502,54
29	219.488,35	179.319,69	170.820,56	187.159,71
30	158.097,05	166.827,49	148.521,03	151.812,34
31	1.275.853,13	1.275.853,13	373.357,29	441.831,02
32	3.170.719,32	3.170.719,32	3.167.501,10	3.170.719,32
33	2.426.664,07	2.709.378,50	2.709.378,50	2.977.925,52
34	1.066.269,82	1.066.269,82	502.645,93	1.066.269,82
35	4.931.665,59	5.429.430,28	5.429.430,28	5.429.430,28
36	3.449.754,40	3.092.796,47	3.092.796,47	3.452.382,83
37	3.778.832,89	4.063.111,08	4.063.111,08	4.063.111,08
38	1.278.366,20	1.278.366,20	1.278.366,20	1.278.366,20
39	785.029,32	785.029,32	785.029,32	785.029,32
40	2.343.962,12	2.324.458,21	2.324.458,21	2.704.129,37
41	2.644.005,59	2.570.020,83	2.570.020,83	1.897.420,78
42	487.333,19	487.333,19	487.333,19	487.333,19
43	986.352,24	985.665,05	985.665,05	986.352,24
44	1.006.025,40	1.209.221,44	1.209.221,44	1.209.221,44
45	492.175,03	492.175,03	492.175,03	492.175,03
46	2.003.900,94	2.004.094,34	2.004.094,34	2.004.094,34
47	3.523.426,07	3.699.069,64	3.699.069,64	3.674.264,14
48	1.696.513,99	2.162.036,21	2.162.036,21	1.481.311,01

A dispersão dos resultados obtidos por cenário, medido pelo coeficiente de variação (desvio padrão/média), foi de 6,47%. A partir dos resultados apresentados na tabela 5.22, verificou-se que o valor médio do “gap” da estratégia combinada *heurística construtiva & “variable neighborhood search”*, é igual a 52,6%, contra 53,7% obtido pela *heurística construtiva & “branch and cut”*. O valor médio da função objetivo da estratégia *heurística construtiva* (isoladamente) tinha ficado 5,4% acima dos valores obtidos pelo “branch and cut”; enquanto que a *heurística construtiva & “branch and cut”* reduziu esta média para -13,4% dos valores obtidos pelo “branch and cut”. O método *heurística construtiva & “variable neighborhood search”* abaixou esta média para -22,6%, lembrando que esta análise foi aplicada para os 39 problemas para os quais se obteve pelo menos uma solução inteira pelo método “branch and cut”.

Figura 5.8 - Evolução do método “variable neighborhood search” aplicado à solução inicial heurística, para o cenário #18

A figura 5.8 exemplifica a melhoria da função objetivo para o cenário 18, testada para cada uma das quatro configurações de vizinhança. Os picos observados nestes gráficos indicam as soluções geradas pela estratégia de diversificação.

5.8 Heurística Construtiva & “Local Branching”

A estratégia de busca em vizinhança “local branching” será testada apenas no modelo de fluxo em rede – formulação compacta, pois foi com este modelo que foram obtidos os melhores resultados com a estratégia “branch and cut”. Este método utilizará como solução inicial a melhor solução heurística³⁷, e terá tempo de processamento de 6 horas. Assim, os dois métodos combinados poderão ser comparados com o desempenho do algoritmo “branch and cut” isoladamente.

A estratégia de busca proposta por Fischetti; Lodi (2003) requer a calibração de alguns parâmetros. Além da duração total, de 6 horas, dois parâmetros influenciam diretamente a exploração da vizinhança: k – indica o tamanho da vizinhança, que é igual a 2^k ; *limite_tempo_nó* – indica o tempo total em que a vizinhança será explorada. Para o primeiro parâmetro, os autores propõem a faixa $10 \leq k \leq 20$; para o segundo parâmetro, são sugeridos valores variando entre 180 e 1200 segundos.

As seguintes combinações de valores serão testadas para os parâmetros (k ; *limite_tempo_nó* [s]): (10; 300), (15; 300), (15; 600), (20; 420). Apesar do método “local branching” poder apresentar resultados ainda melhores explorando-se uma combinação maior destes parâmetros, apenas as 4 configurações foram testadas, por limitação de recursos computacionais. Cada configuração foi aplicada ao conjunto de 48 problemas, cujos resultados são mostrados na tabela 5.23; os resultados de mínimo custo de cada cenário estão destacados em negrito.

³⁷ Caso o método “local branching” seja processado sem uma solução inicial, o algoritmo “branch and cut” será executado até que uma solução inteira seja gerada. Só então a busca em vizinhança iniciará.

Tabela 5.23 - Resultados do método “local branching” aplicado ao modelo de fluxo em rede – formulação compacta, com solução inicial heurística

Cenário	10_300	15_300	15_600	20_420
1	221.478,92	221.478,92	221.478,92	221.478,92
2	320.903,43	320.903,43	320.903,43	320.903,43
3	121.749,50	121.749,50	121.749,50	121.749,50
4	304.807,98	304.807,98	304.807,98	304.807,98
5	979.632,35	979.632,35	979.632,35	979.632,35
6	182.898,78	182.898,78	182.898,78	182.898,78
7	401.906,02	401.906,02	401.906,02	401.906,02
8	629.357,48	629.357,48	629.357,48	629.357,48
9	237.141,21	237.141,21	237.141,21	237.141,21
10	207.867,45	207.867,45	207.867,45	207.867,45
11	1.009.457,92	1.009.457,92	1.009.457,92	1.009.457,92
12	261.365,23	261.365,23	261.365,23	261.365,23
13	417.637,11	416.878,24	413.280,49	272.058,32
14	328.440,66	281.091,50	281.028,71	250.156,93
15	107.623,00	107.576,17	107.576,17	107.623,00
16	1.778.281,79	1.778.332,27	1.778.372,14	1.780.106,02
17	552.335,54	623.419,57	552.411,18	552.317,59
18	606.808,79	605.016,16	612.887,25	581.159,21
19	504.489,25	509.030,62	504.572,14	514.087,29
20	104.791,25	104.860,98	104.767,81	101.841,44
21	205.702,83	207.440,43	208.896,73	197.284,65
22	143.402,61	142.016,23	141.969,62	141.971,68
23	220.686,60	217.024,67	219.936,77	221.999,35
24	305.000,47	316.285,96	312.129,17	302.190,76
25	446.849,82	671.495,87	593.313,73	459.405,41
26	1.674.259,10	1.668.012,29	1.666.185,79	1.665.117,87
27	147.952,81	122.664,94	120.931,71	120.890,24
28	233.100,98	233.239,88	229.253,05	229.315,42
29	171.574,89	169.448,35	131.520,18	129.554,30
30	141.135,98	140.621,86	142.116,03	142.413,03
31	1.032.622,74	321.455,87	420.306,39	1.191.152,90
32	2.674.430,00	3.170.719,32	2.393.678,30	3.042.437,85
33	1.842.895,79	2.656.031,46	1.142.210,20	2.784.675,20
34	965.057,92	151.685,70	144.197,43	322.248,21
35	3.273.185,73	4.203.345,74	4.004.163,79	5.429.430,28
36	2.689.887,17	2.520.446,28	2.969.320,05	3.091.025,42
37	3.743.866,17	3.836.305,61	3.349.010,18	2.850.312,84
38	1.278.366,20	878.327,11	764.206,00	1.086.986,67
39	224.341,90	207.593,50	205.926,98	221.334,69
40	2.704.129,37	1.554.491,90	677.195,37	954.901,64
41	1.713.348,63	2.460.909,33	590.864,31	1.713.601,37
42	136.534,50	137.331,90	132.763,83	131.421,16
43	535.446,39	752.056,71	693.940,32	985.665,05
44	806.357,15	765.648,28	740.323,95	881.336,72
45	475.239,58	396.902,91	389.711,10	473.178,76
46	1.981.193,94	1.504.834,26	1.067.374,82	1.841.051,08
47	3.241.940,22	3.607.631,11	2.512.233,89	2.912.458,56
48	1.740.538,06	2.162.036,21	2.037.988,96	2.162.036,21

A análise destes resultados mostrou que, nos problemas de 10 tarefas, as quatro configurações de vizinhança levaram ao mesmo resultado final. Para os demais problemas houve apenas um caso de empate (cenário 15). Excluindo os cenários de 10 tarefas, observou-se a seguinte incidência do tipo de vizinhança nas melhores soluções: (10; 300): 6 vezes, (15; 300): 5 vezes, (15; 600): 13 vezes, (20; 420): 12 vezes. Em especial, 8 das 13 ocorrências da estrutura (15; 600) foi para os problemas de 40 tarefas.

A dispersão dos resultados obtidos por cenário, medido pelo coeficiente de variação (desvio padrão/média), foi de 12,00%. A partir dos valores apresentados na tabela 5.23, o valor médio do “gap” da estratégia combinada *heurística construtiva & “local branching”* foi calculado em 43,6%, contra 52,6% obtido pela *heurística construtiva & “variable neighborhood search”*, 53,7% da *heurística construtiva & “branch and cut”* e 61,7% da *heurística construtiva*.

O valor médio da função objetivo da estratégia *heurística construtiva* tinha ficado 5,4% acima dos valores obtidos pelo “branch and cut”; para a *heurística construtiva & “branch and cut”*, a média foi 13,4% menor que o “branch and cut”, para a *heurística construtiva & “variable neighborhood search”*, a média foi 22,6% menor e para a *heurística construtiva & “local branching”*, este valor foi de 34,0% menor. A figura 5.9 exemplifica, para o cenário 13, a evolução da função objetivo, testada para cada uma das quatro configurações de vizinhança. Os picos observados nestes gráficos indicam as soluções geradas pela estratégia de diversificação.

Figura 5.9 - Evolução do método “local branching” aplicado a partir de uma solução heurística inicial, para o cenário #18

5.9 Influência das Tarefas de Abastecimento e Troca de Tripulação

O retorno de uma embarcação à base ocorre em função da necessidade de abastecimento ou para troca de tripulação. Além disso, quando uma embarcação é alocada a tarefas de transporte de suprimento, o retorno à base é necessário para carregamento destes suprimentos – caso que não é tratado nesta pesquisa.

A modelagem matemática apresentada no capítulo 3 propôs que o abastecimento e a troca de tripulação fossem tratadas como tarefas iguais às demais, com a ressalva de que apenas a embarcação destinada a ser abastecida ou a trocar de tripulação pudesse atender à tarefa. Assim, as restrições (3.23) a (3.25) do modelo de fluxo em rede – formulação compacta, a seguir, reproduzidas, que tratam do atendimento da demanda e da continuidade do fluxo em cada tarefa, teriam que ter

os parâmetros de compatibilidade tarefa-embarcação - $compat(j,r)$ - ajustados para apenas permitir o fluxo da embarcação designada a ir para a base.

$$\sum_{r \in emb(c)} \sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r compat(j,r) \geq q(j,c) \quad j \in N_2, c \in C \wedge q(j,c) > 0 \quad (3.23)$$

$$\sum_{r \in E} \sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r compat(j,r) = \sum_{c \in C} q(j,c) \quad j \in N_2 \quad (3.24)$$

$$\sum_{\substack{i \in N_1(r) \cup N_2 \\ i \neq j}} x_{ij}^r compat(j,r) = \sum_{\substack{k \in N_2 \cup N_3(r) \\ k \neq j}} x_{jk}^r compat(k,r) \quad r \in E, j \in N_2 \quad (3.25)$$

Para avaliar a eficácia desta representação, foram selecionados alguns cenários para teste (#25 a #30), contendo, cada um, 30 tarefas. Para estes problemas, 10 embarcações foram escolhidas e, para cada uma delas, foi sorteada, arbitrariamente, uma data de retorno à base para abastecimento. A janela de tempo correspondente às tarefas de abastecimento foi de dois dias de antecipação e um dia de atraso com relação à data prevista para o abastecimento. Para o cenário 29, as tarefas de abastecimento criadas estão listadas nas últimas 10 linhas da tabela 5.24, juntamente com as demais tarefas.

Para estes problemas, utilizou-se como procedimento de solução o método “branch and cut” que acompanha o pacote computacional CPLEX 10.0, sendo processado por, no máximo, 12 horas.

A influência das tarefas de abastecimento foi notória, ao reduzir a disponibilidade das embarcações durante o horizonte de planejamento, com consequência direta nos instantes de chegada e de início das tarefas. O efeito da inclusão das tarefas de abastecimento na solução deste cenário pode ser observado na tabela 5.25. É importante destacar que a incorporação das tarefas de abastecimento

aumentou a complexidade do modelo fazendo com que o “gap” final após 12 horas de processamento fosse igual 78,8%, contra 12,6% do modelo sem abastecimento.

Tabela 5.24 – Lista de Tarefas do Cenário #29

Tarefa	Limite Inferior	Data Início	Duração [dias]	Penal. [\$/dia]	Limite Superior
1	10/01/06	15/01/06	14	39.109,00	20/01/06
2	05/02/06	10/02/06	3	86.846,00	15/02/06
3	14/01/06	19/01/06	15	28.537,00	24/01/06
4	11/01/06	16/01/06	22	168.757,00	21/01/06
5	15/01/06	20/01/06	18	152.220,00	25/01/06
6	25/01/06	30/01/06	15	9.466,00	04/02/06
7	01/01/06	06/01/06	7	194.818,00	11/01/06
8	04/01/06	09/01/06	6	267.493,00	14/01/06
9	26/01/06	31/01/06	5	63.276,00	05/02/06
10	01/01/06	06/01/06	10	54.499,00	11/01/06
11	26/01/06	31/01/06	13	165.109,00	05/02/06
12	15/01/06	20/01/06	8	120.449,00	25/01/06
13	25/01/06	30/01/06	15	12.923,00	04/02/06
14	18/01/06	23/01/06	11	160.013,00	28/01/06
15	06/01/06	11/01/06	15	65.210,00	16/01/06
16	01/01/06	06/01/06	19	170.223,00	11/01/06
17	12/01/06	17/01/06	16	53.367,00	22/01/06
18	31/01/06	05/02/06	7	189.384,00	10/02/06
19	18/01/06	23/01/06	15	244.986,00	28/01/06
20	20/01/06	25/01/06	2	187.225,00	30/01/06
21	21/01/06	26/01/06	15	13.377,00	31/01/06
22	27/01/06	01/02/06	2	36.306,00	06/02/06
23	22/01/06	27/01/06	18	3.360,00	01/02/06
24	03/01/06	08/01/06	14	214.677,00	13/01/06
25	04/01/06	09/01/06	16	183.287,00	14/01/06
26	27/01/06	01/02/06	13	129.782,00	06/02/06
27	11/01/06	16/01/06	11	233.325,00	21/01/06
28	15/01/06	20/01/06	10	313.615,00	25/01/06
29	18/01/06	23/01/06	13	159.989,00	28/01/06
30	08/01/06	13/01/06	13	99.128,00	18/01/06
Emb. 1	14/01/06	16/01/06	1	0,00	17/01/06
Emb. 3	25/01/06	27/01/06	1	0,00	28/01/06
Emb. 5	02/01/06	04/01/06	1	0,00	05/01/06
Emb. 9	16/01/06	18/01/06	1	0,00	19/01/06
Emb. 11	14/01/06	16/01/06	1	0,00	17/01/06
Emb. 14	09/01/06	11/01/06	1	0,00	12/01/06
Emb. 16	03/02/06	05/02/06	1	0,00	06/02/06
Emb. 19	01/01/06	03/01/06	1	0,00	04/01/06
Emb. 23	18/01/06	20/01/06	1	0,00	21/01/06
Emb. 26	07/02/06	09/02/06	1	0,00	10/02/06

Tabela 5.25 – Comparação dos Modelos com e sem Abastecimento para o Cenário #29

Com Abastecimento				Sem Abastecimento			
Tarefa	Instante Início	Atraso [dias]	Atraso x Penal	Tarefa	Instante Início	Atraso [dias]	Atraso x Penal
Emb. 19	1,00	0,00	0,00	7	1,44	0,00	0,00
10	1,91	0,00	0,00	16	1,90	0,00	0,00
16	1,92	0,00	0,00	10	1,91	0,00	0,00
Emb. 5	2,00	0,00	0,00	24	3,00	0,00	0,00
7	2,39	0,00	0,00	8	4,00	0,00	0,00
24	3,00	0,00	0,00	25	4,00	0,00	0,00
8	4,00	0,00	0,00	15	6,00	0,00	0,00
25	4,00	0,00	0,00	30	8,00	0,00	0,00
15	6,00	0,00	0,00	1	10,00	0,00	0,00
30	8,00	0,00	0,00	4	11,00	0,00	0,00
Emb. 14	9,00	0,00	0,00	27	11,00	0,00	0,00
1	10,00	0,00	0,00	17	12,00	0,00	0,00
27	11,00	0,00	0,00	3	14,00	0,00	0,00
4	11,94	0,00	0,00	12	15,00	0,00	0,00
Emb. 1	14,00	0,00	0,00	28	15,00	0,00	0,00
Emb. 11	14,00	0,00	0,00	5	17,30	0,00	0,00
28	15,00	0,00	0,00	29	18,00	0,00	0,00
Emb. 9	16,00	0,00	0,00	19	20,20	0,00	0,00
17	16,08	0,00	0,00	14	22,02	0,00	0,00
5	16,09	0,00	0,00	20	24,30	0,00	0,00
12	17,62	0,00	0,00	21	25,49	0,00	0,00
19	18,00	0,00	0,00	23	26,30	0,00	0,00
29	18,00	0,00	0,00	11	26,57	0,00	0,00
Emb. 23	18,00	0,00	0,00	22	27,00	0,00	0,00
14	21,04	0,00	0,00	6	29,07	0,00	0,00
3	22,00	3,00	85.611,00	9	29,20	0,00	0,00
Emb. 3	25,00	0,00	0,00	13	29,58	0,00	0,00
21	25,62	0,00	0,00	26	31,97	0,00	0,00
20	25,72	0,72	134.964,50	18	33,00	0,00	0,00
9	26,00	0,00	0,00	2	40,18	0,00	0,00
23	27,72	0,72	2.422,12				
11	27,99	0,00	0,00				
22	31,79	0,00	0,00				
26	32,58	0,58	74.985,16				
18	33,06	0,00	0,00				
Emb. 16	34,00	0,00	0,00				
13	34,17	4,17	53.932,70				
6	34,71	4,71	44.576,58				
Emb. 26	38,00	0,00	0,00				
2	41,23	0,23	20.292,64				

Para este cenário, as parcelas de custo variável e penalidade são, respectivamente: \$ 129.515,46 e \$ 416.784,70 para o problema com as tarefas de abastecimento; \$ 129.730,18 e \$ 0,00 para o problema sem as tarefas de abastecimento. Embora a inclusão das 10 tarefas adicionais não tenha influenciado a parcela de custo variável, a parcela de penalidade foi significativamente afetada. Os resultados dos demais cenários testados estão listados na tabela 5.26. Em todos os demais casos, observou-se um aumento no custo variável, pelo acréscimo do deslocamento, bem como uma variação na parcela de penalidade.

Tabela 5.26 – Influência das Tarefas de Abastecimento nos Cenários #25 a #28

Cenário 25			
Com Abastecimento		Sem Abastecimento	
Penalidade [\$]	3.080.703,33	Penalidade [\$]	1.787.456,23
Custo Variável [\$]	119.249,72	Custo Variável [\$]	104.263,30
Total [\$]	3.199.953,05	Total [\$]	1.891.719,53
Atraso Médio* [dias]	0,98	Atraso Médio [dias]	0,47
Atraso Médio Todas** [dias]	0,73		
Cenário 26			
Com Abastecimento		Sem Abastecimento	
Penalidade [\$]	1.794.059,50	Penalidade [\$]	1.529.964,08
Custo Variável [\$]	144.091,98	Custo Variável [\$]	130.704,53
Total [\$]	1.938.151,48	Total [\$]	1.660.668,61
Atraso Médio [dias]	0,51	Atraso Médio [dias]	0,43
Atraso Médio Todas [dias]	0,38		
Cenário 27			
Com Abastecimento		Sem Abastecimento	
Penalidade [\$]	1.245.415,47	Penalidade [\$]	32.705,05
Custo Variável [\$]	150.083,56	Custo Variável [\$]	130.153,07
Total [\$]	1.395.499,03	Total [\$]	162.858,12
Atraso Médio [dias]	0,46	Atraso Médio [dias]	0,11
Atraso Médio Todas [dias]	0,34		
Cenário 28			
Com Abastecimento		Sem Abastecimento	
Penalidade [\$]	1.010.367,35	Penalidade [\$]	727.141,69
Custo Variável [\$]	155.373,27	Custo Variável [\$]	138.141,64
Total [\$]	1.165.740,63	Total [\$]	865.283,33
Atraso Médio [dias]	0,44	Atraso Médio [dias]	0,54
Atraso Médio Todas [dias]	0,33		

* - Apenas as tarefas convencionais; ** - Incluindo as tarefas de abastecimento

5.10 Influência da Parcela de Penalidade

Outro aspecto avaliado foi a influência da parcela de penalidade nas soluções dos problemas. Como a ordem de grandeza das penalidades é maior que os valores de custo variável, buscou-se verificar a sensibilidade das respostas do método “branch and cut” com relação ao peso da parcela de penalidade.

Para que a análise ficasse consistente, foram escolhidos problemas de pequeno porte (10 tarefas) para os quais o algoritmo “branch and cut” conseguiria atingir a solução ótima nas várias configurações (pesos) das parcelas de penalidade.

A análise do cenário 11, em que as parcelas de penalidade (em \$/dia) das 10 tarefas são, respectivamente: \$146.027,00; \$120.097,00; \$86.773,00; \$20.085,00; \$300.130,00; \$160.055,00; \$140.102,00; \$60.488,00; \$297.749,00; \$210.713,00, indicou fatos interessantes mostrados na tabela 5.27. Nesta tabela é indicado o peso atribuído à penalidade, isto é, um fator α que multiplica a parcela de penalidade na função objetivo, a seguir reproduzida.

$$\min C = \sum_{j \in N_2} Atraso_j penal(j)\alpha + \sum_{r \in E} \sum_{i \in N_1(r) \cup N_2} \sum_{j \in N_2 \cup N_3(r)} (cv(r)tempo(i, j, r)x_{ij}^r) \quad (5.1)$$

A tabela 5.27 também mostra as parcelas de custo variável e penalidade da solução ótima (“gap” inferior a 1%) para os testes realizados. A coluna “penalidade equivalente” estima o valor da penalidade para o caso real, em que a solução obtida é mantida e os atrasos são ponderados pela penalidade integral. Por último, tem-se o custo total equivalente, calculado pela soma da parcela de custo variável com a parcela de “penalidade equivalente”. Exceto para os casos em que $\alpha=0\%$ e $\alpha=1\%$, o peso das parcelas de penalidade é suficientemente grande para gerar, aproximadamente, o mesmo valor da função objetivo – lembrando que, para estes

casos, as diferenças observadas em “penalidade equivalente” são justificadas pelo critério de parada de “gap” $\leq 1\%$.

Tabela 5.27 – Influência da Parcela de Penalidade para o Cenário #11

Peso Penalidade	Custo Variável [\$]	Penalidade [\$]	Total [\$]	Penalidade Equivalente [\$]	Total Equivalente [\$]
0%	61.650,39	0,00	61.650,39	2.822.822,87	2.884.473,26
1%	65.130,52	11.769,59	76.900,11	1.176.958,66	1.242.089,18
5%	69.373,18	47.031,26	116.404,44	940.625,29	1.009.998,47
10%	69.562,42	94.200,48	163.762,90	942.004,78	1.011.567,21
20%	69.373,18	189.125,19	258.498,37	945.625,95	1.014.999,13
40%	69.784,60	378.250,38	448.034,98	945.625,95	1.015.410,55
60%	72.829,35	566.408,14	639.237,49	944.013,56	1.016.842,91
80%	69.355,25	753.603,83	822.959,08	942.004,78	1.011.360,03
100%	65.130,52	940.625,29	1.005.755,81	940.625,29	1.005.755,81

A mesma situação foi constatada em outros cenários. A tabela 5.28 mostra o efeito do peso α na função objetivo para outros 3 cenários, em que todas as soluções obtidas foram as ótimas, segundo o critério de parada.

5.11 Influência das Desigualdades Válidas

O método “branch and bound” empregado nesta pesquisa é a versão presente no pacote computacional CPLEX 10.0. Conforme discutido na revisão bibliográfica, o método “branch and cut” consiste na geração de uma solução utilizando-se do método “branch and bound” associada à verificação, em cada nó da árvore, de quais desigualdades válidas foram violadas na versão relaxada do problema. O uso de desigualdades válidas leva a uma formulação matemática mais restrita, permitindo a geração de limitantes inferiores de melhor qualidade.

Tabela 5.28 – Influência da Parcela de Penalidade para os Cenários #1, #3 e #7

Cenário #1

Peso Penalidade	Custo Variável [\$]	Penalidade [\$]	Total [\$]	Penalidade Equivalente [\$]	Total Equivalente [\$]
0%	39.856,11	0,00	39.856,11	910.917,21	950.773,32
20%	45.356,97	35.894,51	81.251,48	179.472,57	224.829,54
40%	46.903,75	70.208,07	117.111,82	175.520,17	222.423,93
60%	46.942,93	104.829,89	151.772,82	174.716,48	221.659,42
80%	46.903,75	140.416,14	187.319,89	175.520,17	222.423,93
100%	46.903,75	175.520,17	222.423,93	175.520,17	222.423,93

Cenário #3

Peso Penalidade	Custo Variável [\$]	Penalidade [\$]	Total [\$]	Penalidade Equivalente [\$]	Total Equivalente [\$]
0%	48.162,95	0,00	48.162,95	1.495.488,63	1.543.651,57
20%	60.280,39	12.402,66	72.683,05	62.013,32	122.293,71
40%	60.286,00	24.805,33	85.091,33	62.013,32	122.299,32
60%	60.280,39	37.207,99	97.488,38	62.013,32	122.293,71
80%	60.280,39	49.610,66	109.891,04	62.013,32	122.293,71
100%	60.283,39	62.013,32	122.296,71	62.013,32	122.296,71

Cenário #7

Peso Penalidade	Custo Variável [\$]	Penalidade [\$]	Total [\$]	Penalidade Equivalente [\$]	Total Equivalente [\$]
0%	32.567,83	0,00	32.567,83	4.487.978,37	4.520.546,20
20%	41.300,94	72.128,81	113.429,75	360.644,03	401.944,97
40%	41.300,94	144.257,61	185.558,55	360.644,03	401.944,97
60%	41.312,16	215.981,19	257.293,35	359.968,65	401.280,80
80%	41.349,18	288.515,22	329.864,41	360.644,03	401.993,21
100%	41.300,94	360.644,03	401.944,97	360.644,03	401.944,97

O maior desafio para a resolução de um problema de programação inteira é a proposição de desigualdades válidas que levem à definição da envoltura do espaço de solução, pois estas são específicas a cada tipo de problema, e difíceis de serem obtidas. Os softwares de otimização trabalham com um conjunto predefinido de desigualdades genéricas, ou seja, que se aplicam a diversas classes de problemas, cujo melhor exemplo é a família de desigualdades proposta por Chvatal-Gomory.

Para a presente pesquisa, foram conduzidos alguns testes com as desigualdades válidas que acompanham o pacote CPLEX. A proposta destes testes é

de verificar se haveria ganho de desempenho utilizando ou suspendendo o uso de algumas destas desigualdades para a classe de problemas testada. Cabe ressaltar que quaisquer conclusões a respeito da aplicabilidade destas desigualdades se limitam aos casos testados e possuem caráter empírico, isto é, não são consequência de uma análise da estrutura poliédrica da classe de problema testada.

O pacote computacional CPLEX aplica, de maneira sistematizada, as famílias de desigualdades válidas em cada nó da árvore “branch and bound”, para os problemas relaxados. Há algumas formas de testar a eficácia do uso destas famílias. Uma delas é impedir que os cortes sejam adicionados, ou seja, o método “branch and bound” é empregado na sua forma clássica. A outra forma, é habilitar progressivamente cada família de desigualdades e avaliar o efeito que isto causa, na geração de limitantes inferiores de melhor qualidade – lembrando que isto causa um acréscimo no tempo de processamento em cada nó. A estratégia usada foi a de deixar o CPLEX adicionar as desigualdades que julgasse necessário e, em cada teste conduzido, uma família de desigualdades era desabilitada. Assim, o CPLEX incorporaria todas as famílias de desigualdades que entendesse ser importante, menos aquela que estivesse desabilitada.

Para conduzir estes testes, 5 problemas foram escolhidos e processados por uma hora cada. Onze testes foram aplicados para cada um dos 5 problemas. No primeiro teste, nenhuma desigualdade foi desativada (configuração 0, da tabela 5.29). Nas demais configurações (1 a 9), a família indicada na tabela 5.29 foi desativada. Por último, a configuração 10 consistiu em desabilitar mais de uma desigualdade (Cortes por Incompatibilidade e a Desigualdades de Cobertura de Fluxo), cuja desativação tinham causado os melhores resultados nestes testes preliminares.

Tabela 5.29 – Configurações de Desigualdades Testadas

Configuração	Descrição
0	Todas famílias habilitadas
1	Cortes por Incompatibilidade
2	Desigualdades de Cobertura
3	Desigualdades Disjuntivas
4	Desigualdades de Cobertura de Fluxo
5	Desigualdade por Cobertura de Caminhos de Fluxo
6	Desigualdade Válida de Chvátal-Gomory
7	Desigualdade de Cobertura GUB
8	Cortes de Limitantes Implícitos
9	Desigualdade Válida para Arredondamento em Programação Inteira Mista
10	Cortes por Incompatibilidade & Desigualdades de Cobertura de Fluxo

A tabela 5.30 lista os resultados obtidos para cada problema e para cada configuração, mostrando que não há uma regra geral que seja válida sempre. Contudo, é possível verificar que, desabilitando a família 1 (cortes por incompatibilidade) e a família 4 (desigualdade de cobertura de fluxo), todos os problemas, exceto o #4, obtiveram melhores respostas. Assim, optou-se por desabilitar estas 2 famílias simultaneamente, levando à definição da configuração 10. Esta configuração foi a adotada nos diversos processamentos computacionais do método “branch and cut”, previamente descritos nas seções 5.2, 5.3, 5.6 a 5.10.

5.12 “Branch and Cut” aplicado ao Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão II

O algoritmo “branch & cut” que acompanha o pacote computacional CPLEX 10.0 foi aplicado ao conjunto de 48 problemas, usando a formulação matemática do Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão II. Cada problema foi processado por 12 horas em um

computador Pentium 4, 2.4 GHz, 512 Mb RAM, HD local PATA 40 Gb. Os resultados estão apresentados na tabela 5.31.

Tabela 5.30 – Testes com as Desigualdades Válidas

Problema #1			
Configuração	Número Soluções	F.O. [\$] - 1a. Sol.	F.O. [\$] - Últ. Sol.
0	12	3.807.314,17	2.755.416,69
1	3	3.100.728,46	1.142.190,28
2	12	3.807.314,17	2.755.416,69
3	12	3.807.314,17	2.755.416,69
4	6	4.021.511,12	1.540.040,18
5	12	3.807.314,17	2.755.416,69
6	6	8.472.565,92	2.049.113,03
7	12	3.807.314,17	2.755.416,69
8	3	4.801.983,36	4.132.428,76
9	12	3.807.314,17	2.755.416,69
10	3	3.249.989,44	2.202.357,34
Problema #2			
Configuração	Número Soluções	F.O. [\$] - 1a. Sol.	F.O. [\$] - Últ. Sol.
0	3	2.238.779,97	1.977.449,78
1	4	4.190.529,20	1.936.413,37
2	3	2.238.779,97	1.977.449,78
3	3	2.238.779,97	1.977.449,78
4	5	2.838.311,36	1.331.909,44
5	3	2.238.779,97	1.977.449,78
6	3	2.943.948,13	2.328.087,82
7	3	2.238.779,97	1.977.449,78
8	3	5.820.467,75	3.673.315,01
9	3	2.238.779,97	1.977.449,78
10	6	3.821.243,47	1.161.110,25
Problema #3			
Configuração	Número Soluções	F.O. [\$] - 1a. Sol.	F.O. [\$] - Últ. Sol.
0	18	6.610.649,18	549.925,65
1	7	4.892.156,10	171.700,63
2	18	6.610.649,18	549.925,65
3	18	6.610.649,18	549.925,65
4	8	4.375.166,25	255.846,96
5	18	6.610.649,18	549.925,65
6	12	4.254.886,40	700.199,90
7	18	6.610.649,18	549.925,65
8	13	5.239.354,83	606.828,69
9	18	6.610.649,18	549.925,65
10	9	4.654.672,78	167.415,58

Tabela 5.30 – Testes com as Desigualdades Válidas (Continuação)

Problema #4			
Configuração	Número Soluções	F.O. [\$] - 1a. Sol.	F.O. [\$] - Últ. Sol.
0	11	3.605.782,99	1.894.700,38
1	14	4.365.963,92	2.582.984,96
2	11	3.605.782,99	1.894.700,38
3	11	3.605.782,99	1.894.700,38
4	4	3.278.573,46	2.977.589,25
5	11	3.605.782,99	1.894.700,38
6	9	5.022.367,98	2.775.127,48
7	11	3.605.782,99	1.894.700,38
8	6	5.524.372,80	3.270.537,20
9	11	3.605.782,99	1.894.700,38
10	15	3.865.359,63	2.773.999,34
Problema #5			
Configuração	Número Soluções	F.O. [\$] - 1a. Sol.	F.O. [\$] - Últ. Sol.
0	7	5.558.867,75	2.739.346,57
1	10	6.623.649,01	2.328.384,33
2	7	5.558.867,75	2.739.346,57
3	7	5.558.867,75	2.739.346,57
4	7	5.659.733,50	2.622.866,66
5	7	5.558.867,75	2.739.346,57
6	6	7.310.432,37	2.516.403,63
7	7	5.558.867,75	2.739.346,57
8	4	8.311.562,10	4.740.985,23
9	7	5.558.867,75	2.739.346,57
10	6	5.395.913,74	2.056.114,15

Além do critério de parada por tempo (12 horas), o algoritmo encerra quando o “gap”, definido como a distância relativa da melhor estimativa da função objetivo (“bound”) em relação à função objetivo melhor solução obtida (f_{opt}) for menor ou igual a 1%: $gap = (f_{opt} - bound) / f_{opt} \leq 1\%$.

A análise dos resultados³⁸ contidos da tabela 5.31 indica que em 46 dos 48 problemas (96%) obteve-se pelo menos uma solução viável, sendo que em 25 problemas a solução obtida foi a ótima, segundo o critério de parada (“gap” $\leq 1\%$). O “gap” médio das 46 soluções geradas foi de 43,3%.

³⁸ Cenários com mesmo número de tarefas podem ter uma quantidade de variáveis de decisão e de restrições diferentes em função da demanda de embarcações por classe das tarefas que compõem cada cenário.

Tabela 5.31 - “Branch and cut” aplicado ao Modelo de Programação de Frota em Escala de Tempo Discretizada, sem Deslocamento – Versão II

Cenário	Tarefas	Tempo[s]	#Var.	#Bin.	#Restr.	#Sol.	F.Obj. [\$]	Bound [\$]	Nós	GAP
1	10	0,96	5.632	222	2.882	0	62.367,00	62.367,00	0	0,0%
2	10	27,49	5.638	228	2.954	7	264.628,00	264.628,00	79	0,0%
3	10	316,13	6.094	234	3.170	5	0,00	0,00	1270	0,0%
4	10	8,20	6.094	234	3.404	2	123.210,00	123.210,00	25	0,0%
5	10	7,92	5.632	222	3.146	2	459.448,00	459.448,00	10	0,0%
6	10	27.453,40	7.474	264	4.194	15	8.830,00	8.830,00	56620	0,0%
7	10	247,83	5.206	246	3.042	11	170.903,00	170.903,00	1245	0,0%
8	10	1.498,91	5.675	265	3.398	8	282.650,00	282.650,00	6015	0,0%
9	10	10,33	6.123	263	3.547	1	155.112,00	155.112,00	21	0,0%
10	10	175,53	5.660	250	3.470	3	24.170,00	24.170,00	519	0,0%
11	10	1,85	6.580	270	4.040	1	840.612,00	840.612,00	1	0,0%
12	10	177,19	7.036	276	4.265	2	0,00	0,00	370	0,0%
13	20	1.983,13	20.465	645	9.516	5	0,00	0,00	510	0,0%
14	20	2.968,18	21.397	677	10.220	6	0,00	0,00	722	0,0%
15	20	43.200,50	25.077	757	11.980	8	21.861,00	0,00	6475	100,0%
16	20	23.170,60	25.079	759	13.141	18	1.133.809,00	1.128.892,25	2363	0,4%
17	20	43.200,60	26.918	798	14.078	6	1.116.128,00	0,00	4422	100,0%
18	20	43.200,20	28.756	836	15.042	9	238.208,00	0,00	3510	100,0%
19	20	1.853,71	20.553	733	11.496	6	43.961,00	43.961,00	413	0,0%
20	20	43.200,30	21.471	751	11.738	10	573.938,00	0,00	10105	100,0%
21	20	11.477,50	25.156	836	13.681	24	0,00	0,00	2592	0,0%
22	20	1.113,57	23.316	796	14.125	4	15.856,00	15.856,00	180	0,0%
23	20	43.200,60	26.988	868	16.166	7	595.678,00	0,00	5160	100,0%
24	20	43.200,20	28.836	916	17.491	9	280.652,00	8.048,00	7644	97,1%
25	30	43.200,50	42.447	1.017	15.192	3	1.491.935,00	0,00	7061	100,0%
26	30	43.200,40	47.937	1.107	17.310	2	1.776.258,00	847.364,00	6804	52,3%
27	30	1.834,64	49.767	1.137	17.943	5	0,00	0,00	146	0,0%
28	30	7.345,12	49.762	1.132	19.347	9	0,00	0,00	480	0,0%
29	30	43.200,80	53.430	1.200	21.059	4	330.218,00	0,00	1821	100,0%
30	30	2.884,42	57.083	1.253	22.346	1	0,00	0,00	38	0,0%
31	30	43.200,40	38.908	1.078	16.467	1	8.810.350,00	0,00	5019	100,0%
32	30	43.200,20	42.569	1.139	18.067	4	1.119.676,00	0,00	4133	100,0%
33	30	43.200,90	48.057	1.227	20.378	7	972.973,00	196.286,00	3369	79,8%
34	30	43.200,60	46.229	1.199	21.200	6	2.124.030,00	0,00	2025	100,0%
35	30	43.200,70	53.550	1.320	24.539	2	6.079.251,00	176.754,00	3472	97,1%
36	30	43.200,00	57.208	1.378	26.128	0		0,00		
37	40	43.200,30	67.350	1.310	19.120	2	1.632.432,00	572.100,00	8176	65,0%
38	40	43.200,40	76.467	1.427	21.775	4	246.913,00	0,00	4152	100,0%
39	40	2.625,37	82.549	1.509	23.366	3	0,00	0,00	118	0,0%
40	40	14.889,00	79.516	1.476	24.926	4	0,00	0,00	427	0,0%
41	40	43.200,70	82.553	1.513	25.688	3	985.612,00	0,00	2801	100,0%
42	40	43.201,30	97.750	1.710	30.632	1	152.040,00	0,00	466	100,0%
43	40	43.201,20	70.534	1.494	23.499	1	1.051.263,00	0,00	2782	100,0%
44	40	43.200,70	76.609	1.569	25.175	2	902.142,00	0,00	2034	100,0%
45	40	2.375,78	82.698	1.658	27.470	1	0,00	0,00	73	0,0%
46	40	43.200,00	82.693	1.653	29.468	0		0,00		
47	40	35.098,10	82.686	1.646	29.522	3	50.834,00	50.834,00	1172	0,0%
48	40	43.201,40	97.902	1.862	35.272	3	1.020.573,00	0,00	937	100,0%

A tabela 5.32 agrupa os resultados da tabela 5.31 por quantidade de tarefas, indicando para quantos problemas pelo menos uma solução viável foi gerada e quantas destas soluções são ótimas, segundo o critério de parada (“gap” $\leq 1\%$). A tabela também mostra os valores médios do número total de variáveis, de variáveis binárias, de restrições, o número médio de soluções geradas e a quantidade média de nós da árvore “branch and cut” gerados no decorrer do processamento, e o valor médio do “gap”.

Tabela 5.32 - Resultados agrupados por número de tarefas

# Tarefas	# Problemas c/ Solução Inteira	# Problemas c/ Solução Ótima	# Médio Variáveis	# Médio Variáveis Binárias	# Médio Restrições	# Médio Soluções Geradas	# Médio Nós	GAP Médio
10	12	12	6.070,3	247,8	3.459,3	4,8	5.514,6	0,00%
20	12	6	24.501,0	781,0	13.222,8	9,3	3.674,7	49,80%
30	11	3	48.912,3	1.182,3	19.998,0	3,7	3.124,4	66,29%
40	11	4	81.608,9	1.568,9	26.326,1	2,3	2.103,5	60,45%

O sucesso da aplicação do método “branch and cut”, comprovada pelo alto número de problemas resolvidos e pela qualidade dos resultados, deve-se ao reduzido número médio de variáveis binárias e de restrições necessárias para representar o problema, quando comparado à formulação de fluxo em rede (versões completa e compacta). Entretanto, é importante ressaltar que o modelo não contempla os tempos de deslocamento das embarcações entre uma tarefa e a subsequente.

As variáveis binárias que compõem a solução deste modelo, presentes nos arquivos de saída do algoritmo “branch and cut” caracterizam de maneira completa a solução, pois indicam o instante de início de cada tarefa, bem como o conjunto de embarcações designada a cada tarefa. A tabela 5.33 lista quais variáveis binárias são iguais a 1 na solução do problema do cenário 5, apresentado anteriormente na seção 5.3.

Tabela 5.33 – Especificação das variáveis binárias com valor igual a 1 na melhor solução gerada pelo método “branch and cut” para o cenário #5

w_{g_j}	x_{r_j}	x_{r_j}	x_{r_j}
w_{6_1}	x_{1_4}	$x_{5_{10}}$	x_{10_2}
w_{12_2}	x_{1_7}	x_{6_1}	x_{10_4}
w_{20_3}	x_{1_9}	x_{6_2}	x_{10_7}
w_{35_4}	x_{2_4}	x_{7_5}	x_{11_3}
w_{39_5}	x_{2_7}	$x_{7_{10}}$	x_{12_1}
w_{50_6}	x_{2_9}	x_{8_1}	x_{12_6}
w_{66_7}	x_{3_8}	x_{8_2}	
w_{77_8}	x_{4_3}	x_{9_1}	
w_{83_9}	$x_{4_{10}}$	x_{9_6}	
$w_{98_{10}}$	x_{5_5}	x_{9_9}	

A primeira coluna refere-se à variável w_j^g que assume valor 1 se a tarefa j tiver início no período de tempo g , cuja numeração é seqüencial e única para todas as tarefas. Assim, a primeira tarefa, cuja janela de tempo estende-se do instante 12 até o instante 22 (conforme representado na tabela 5.34), será representada pelo índice g variando de 1 a 11. Os possíveis instantes de início da segunda tarefa, que variam de 1 a 8, serão representados pelo índice g variando de 12 a 19, e assim sucessivamente. A correspondência entre o índice g e o instante que ele efetivamente representa é feita por meio do parâmetro $inst(g)$. Por exemplo, $inst(12)=1$ e $inst(19)=8$, indicando, para a tarefa 2, os limites inferior e superior da janela de tempo. A tabela 5.34 apresenta, para cada tarefa, sua respectiva faixa de valores do índice g , e quais instantes de tempo que estes índices representam.

Tabela 5.34 – Possíveis instantes de início para as tarefas do cenário #5

Tarefa	LI	LS	Instantes			
1	12	22	$inst(1)=12$	$inst(2)=13$	...	$inst(11)=22$
2	1	8	$inst(12)=1$	$inst(13)=2$	...	$inst(19)=8$
3	1	8	$inst(20)=1$	$inst(21)=2$	...	$inst(27)=8$
4	7	17	$inst(28)=7$	$inst(29)=8$	...	$inst(38)=17$
5	1	11	$inst(39)=1$	$inst(40)=2$	...	$inst(49)=11$
6	4	14	$inst(50)=4$	$inst(51)=5$	...	$inst(60)=14$
7	15	25	$inst(61)=15$	$inst(62)=16$	...	$inst(71)=25$
8	1	11	$inst(72)=1$	$inst(73)=2$	...	$inst(82)=11$
9	1	9	$inst(83)=1$	$inst(84)=2$	...	$inst(91)=9$
10	12	22	$inst(92)=12$	$inst(93)=13$	...	$inst(102)=22$

A tabela 5.35 indica os instantes de tempo em que cada tarefa iniciou, de acordo com as variáveis de decisão apresentadas na tabela 5.33.

Tabela 5.35 – Instantes de início para as tarefas do cenário #5

Tarefa	LI	LS	Faixa Valores g			w_g_j	Instante
1	12	22	1	a	11	w_6_1	17
2	1	8	12	a	19	w_12_2	1
3	1	8	20	a	27	w_20_3	1
4	7	17	28	a	38	w_35_4	14
5	1	11	39	a	49	w_39_5	1
6	4	14	50	a	60	w_50_6	4
7	15	25	61	a	71	w_66_7	20
8	1	11	72	a	82	w_77_8	6
9	1	9	83	a	91	w_83_9	1
10	12	22	92	a	102	w_98_10	18

Com os instantes de início e as embarcações definidas, elaborou-se, para as tarefas, um gráfico de Gantt similar ao apresentado na figura 5.2, conforme mostra a figura 5.10. De maneira análoga, cada linha, cuja escala é o tempo, representa uma tarefa. Os limites inferior e superior da janela de tempo são representados pelos colchetes, e a barra azul, que tem início no instante correspondente à data de início da tarefa, representa a duração da tarefa. A figura 5.11 contém o gráfico de Gantt das embarcações. Cada linha representa uma embarcação e destaca a execução de cada tarefa em uma barra azul cujo comprimento é proporcional à duração da tarefa. Em cima da barra azul, consta o número da tarefa.

Figura 5.10 - Gráfico de Gantt das tarefas – cenário #5

Figura 5.11 – Gráfico de Gantt das embarcações contendo a sequência de tarefas realizadas – cenário #5

A análise do gráfico da figura 5.10 mostra que duas tarefas atrasaram, a saber: tarefa 4, por dois dias, e a tarefa 10, por um dia. Estes atrasos, ponderados pelas penalidades, totalizam: $2 \times \$ 227.398,00 + 1 \times \$ 4.652,00$, perfazendo um total de \$ 459.448,00, conforme consta na tabela 5.24.

Em relação ao resultado obtido pelo modelo de fluxo em rede – formulação compacta, a tabela 5.36 compara as duas soluções. Fica claro que a não incorporação dos tempos de viagem no modelo de tempo discreto influencia, inclusive, na escolha das embarcações, levando a resultados distintos, como esperado.

Tabela 5.36 – Comparação da solução para o cenário #5, para os modelos de fluxo em rede e de programação de frota em escala de tempo discretizada

Tarefa	Modelo de Fluxo em Rede*					Modelo de Tempo Discreto				
	Emb. Alocadas				Data Início	Emb.Alocadas				Data Início
1	4	6	11	12	17,28	6	8	9	12	17,00
2	5	6	7		1,88	6	8	10		1,00
3	8	9			1,75	4	11			1,00
4	5	7	10		15,81	1	2	10		14,00
5	1	2			1,77	5	7			1,00
6	11	12			4,00	9	12			4,00
7	1	2	9		19,21	1	2	10		20,00
8	3				5,78	3				6,00
9	3	4	10		2,01	1	2	9		1,00
10	7	8	10		22,00	4	5	7		18,00

* Não foram incluídas as tarefas fictícias

5.13 “Branch and Cut” aplicado ao Modelo de Programação de Frota em Escala de Tempo Discretizada, Com Deslocamento

O algoritmo “branch and cut” também foi testado para o Modelo de Programação de Frota em Escala de Tempo Discretizada, com Deslocamento, isto é, contemplando os fluxos das embarcações e os respectivos tempos de viagem. Seguindo o padrão adotado ao longo dos testes dos diversos métodos, limitou-se o

tempo de processamento em 12 horas, podendo este ser interrompido quando o “gap” caísse para um patamar inferior a 1%.

Para este modelo, é importante definir a unidade de tempo padrão para representação dos tempos de viagem das embarcações. Como os tempos envolvidos no deslocamento entre os locais onde as tarefas são executadas são, em geral, inferiores a um dia, poderia ser adotada a representação horária da escala de tempo, isto é, cada dia sendo subdividido em 24 períodos de uma hora. Contudo, face às imprecisões inerentes à especificação das datas de início das tarefas, é desnecessário um cálculo refinado dos tempos de viagem. Além disso, é importante considerar que o número de variáveis binárias é diretamente afetado pelo número de períodos. Por exemplo, cada tarefa j inicia em um instante g , definido por w_j^g ; a subdivisão de um dia em vários períodos multiplicaria o número de variáveis binárias w_j^g pelo número de sub-períodos criados.

Assim, foram testadas subdivisões de um dia em 3, 4 e 5 períodos – designados de *fatores de discretização* – com o objetivo de avaliar o impacto destas unidades de tempo nas soluções geradas. Os resultados obtidos são apresentados nas tabelas 5.37 a 5.39, apenas para os cenários de 10 tarefas. Para os demais cenários, nenhuma solução foi obtida.

A tabela 5.40 agrupa os resultados destas três tabelas, mostrando a redução do número de problemas resolvidos à otimalidade com o aumento do número de variáveis binárias e de restrições, ou seja, com o aumento da discretização.

Tabela 5.37 – Resultados do método “branch and cut” aplicado ao modelo de programação de frota em escala de tempo discretizada – fator de discretização 3

Cenário	Tempo [s]	# Var.	# Bin.	# Restr.	# Sol.	F. Obj. [\$]	Bound [\$]	Nós	GAP
1	1.298,89	19.280	1.738	11.318	10	151.043,33	149.543,29	617	0,99%
2	3.737,45	19.298	1.756	11.534	12	318.032,99	314.856,08	1.039	1,00%
3	43.200,30	20.868	1.865	12.413	7	60.925,03	58.441,68	17.800	4,08%
4	2.023,37	20.868	1.865	13.115	10	307.490,01	304.802,46	410	0,87%
5	7.543,72	19.280	1.738	12.110	13	766.367,02	758.730,97	2.502	1,00%
6	12.698,10	25.614	2.228	16.178	6	161.414,39	159.801,09	3.979	1,00%
7	43.200,30	17.800	1.719	11.567	10	327.315,03	35.378,35	10.567	89,19%
8	43.200,30	19.409	1.867	12.866	15	537.375,32	337.888,88	8.466	37,12%
9	19.105,00	20.955	1.952	13.544	7	218.058,71	215.879,51	6.020	1,00%
10	43.200,40	19.364	1.822	13.082	12	125.724,30	48.691,68	15.155	61,27%
11	331,64	22.528	2.064	15.254	2	908.178,75	900.158,86	69	0,88%
12	43.200,30	24.098	2.173	16.160	7	50.301,68	41.374,53	11.366	17,75%

Tabela 5.38 - Resultados do método “branch and cut” aplicado ao modelo de programação de frota em escala de tempo discretizada – fator de discretização 4

Cenário	Tempo [s]	# Var.	# Bin.	# Restr.	# Sol.	F. Obj. [\$]	Bound [\$]	Nós	GAP
1	43.200,30	24.772	1.830	14.150	7	146.590,50	145.106,13	20.844	1,01%
2	4.615,83	24.796	1.854	14.438	10	318.911,75	315.754,42	720	0,99%
3	43.200,50	26.812	1.959	15.533	11	63.433,75	60.991,25	11.476	3,85%
4	3.301,70	26.812	1.959	16.469	15	307.373,75	304.745,83	598	0,85%
5	30.440,40	24.772	1.830	15.206	17	768.476,25	760.823,63	5.273	1,00%
6	24.357,70	32.908	2.322	20.322	9	154.838,75	153.291,25	4.256	1,00%
7	43.200,40	22.876	1.845	14.559	7	339.281,50	26.172,50	7.847	92,29%
8	43.200,60	24.944	2.002	16.214	18	543.217,00	135.530,40	5.625	75,05%
9	43.200,50	26.928	2.075	17.041	18	220.620,75	215.363,25	7.212	2,38%
10	43.200,40	24.884	1.942	16.502	12	119.674,50	47.203,75	8.766	60,56%
11	7.801,25	28.948	2.184	19.244	16	912.497,00	903.382,63	1.908	1,00%
12	43.200,60	30.964	2.289	20.375	11	57.637,50	43.466,88	5.732	24,59%

Tabela 5.39 - Resultados do método “branch and cut” aplicado ao modelo de programação de frota em escala de tempo discretizada – fator de discretização 5

Cenário	Tempo [s]	# Var.	# Bin.	# Restr.	# Sol.	F. Obj. [\$]	Bound [\$]	Nós	GAP
1	43.200,50	30.264	1.922	16.982	13	152.337,20	150.237,20	14.743	1,38%
2	11.708,40	30.294	1.952	17.342	14	321.122,00	318.236,00	1.451	0,90%
3	43.200,40	32.756	2.053	18.653	8	60.878,00	58.010,80	8.067	4,71%
4	3.956,84	32.756	2.053	19.823	7	309.696,00	306.637,00	378	0,99%
5	25.600,90	30.264	1.922	18.302	10	720.993,40	713.784,24	3.613	1,00%
6	43.200,70	40.202	2.416	24.466	7	170.083,80	72.286,60	2.710	57,50%
7	43.200,70	27.952	1.971	17.551	11	1.062.593,20	27.866,87	4.342	97,38%
8	43.200,40	30.479	2.137	19.562	7	627.046,00	333.567,70	3.428	46,80%
9	43.200,80	32.901	2.198	20.538	12	221.930,00	162.710,40	3.925	26,68%
10	43.200,50	30.404	2.062	19.922	12	132.641,20	49.149,00	6.269	62,95%
11	2.039,02	35.368	2.304	23.234	4	911.827,00	902.744,29	204	1,00%
12	43.200,60	37.830	2.405	24.590	6	58.315,00	42.956,59	3.956	26,34%

Tabela 5.40 - Resultados agrupados por fator de discretização

Fator Discret.	# Problemas c/ Solução Inteira	# Problemas c/ Solução Ótima (gap<1%)	# Médio Variáveis	# Médio Variáveis Binárias	# Médio Restrições	# Médio Soluções Geradas	# Médio Nós	GAP Médio
3	12	7	20.780,2	1.898,9	13.261,8	9,3	6.499,2	18,01%
4	12	5	26.701,3	2.007,6	16.671,1	12,6	6.688,1	22,05%
5	12	4	32.622,5	2.116,3	20.080,4	9,3	4.423,8	27,30%

A tabela 5.41 apresenta os resultados do método “branch and cut” aplicado ao cenário 5 para os três fatores de discretização, indicando o instante de início e as embarcações designadas a cada tarefa. Embora o conjunto de embarcações que executou cada tarefa tenha sido diferente nos três casos, percebe-se que a priorização no atendimento das tarefas foi a mesma, tendo em vista que as mesmas tarefas atrasaram. Para o cenário 5, verificou-se que a discretização do tempo teve impactos no custo total, quando comparado com a formulação de fluxo em rede. O uso de variáveis reais para calcular os tempos de viagem fez com que a tarefa 1 tivesse um atraso de 0,28 dias, a tarefa 4 atrasasse 3,81 dias e a tarefa 10, 5 dias (ver tabela 5.12). Em função da elevada penalidade da tarefa 4 (\$ 227.398,00), o acréscimo de 0,81 dias quando comparado com a resposta do fator de discretização 4, introduz uma diferença de \$ 184.192,00 aproximadamente.

Tabela 5.41 – Detalhamento das soluções, por fator de discretização, para o modelo de programação de frota em escala tempo discretizada, para o cenário #5

Tarefa	Fator 3						Fator 4						Fator 5					
	Embarcações Alocadas				Início	Atraso	Embarcações Alocadas				Início	Atraso	Embarcações Alocadas				Início	Atraso
1	4	5	9	11	17,00	0,00	1	8	11	12	17,00	0,00	5	7	9	10	17,00	0,00
2	2	4	11		2,00	0,00	1	4	12		2,00	0,00	6	7	9		1,80	0,00
3	3	10			1,67	0,00	5	9			2,25	0,00	8	12			2,20	0,00
4	2	6	12		15,00	3,00	3	4	10		15,00	3,00	1	6	11		14,80	2,80
5	7	8			1,67	0,00	2	6			3,00	0,00	3	4			1,80	0,00
6	5	9			4,00	0,00	8	11			4,00	0,00	5	10			4,00	0,00
7	7	8	10		19,67	0,00	2	6	9		20,00	0,00	3	4	12		20,00	0,00
8	1				6,00	0,00	7				6,00	0,00	2				6,00	0,00
9	1	6	12		2,00	0,00	3	7	10		2,00	0,00	1	2	11		2,00	0,00
10	2	3	12		21,00	4,00	3	4	10		21,25	4,25	1	6	8		21,00	4,00

5.14 “Feasibility Pump” aplicado ao Modelo de Fluxo em Rede - Compacto

Aplicou-se o método “feasibility pump” aos modelos de fluxo em rede – versão compacta, com o intuito de verificar a eficácia desta técnica na geração de uma solução viável para cada cenário. Diferentemente das heurísticas construtivas, que possuem um tempo de processamento igual a 6 horas, o método “feasibility pump” encerra quando a primeira solução viável é gerada, exceto se o estágio 3 da heurística for acionado – neste caso, o tempo de limite de processamento deste estágio encerra-se em 6 horas.

O algoritmo utilizado nesta pesquisa é o proposto por Achterberg; Berthold (2005), o qual requer a definição de um conjunto de parâmetros. Estes estão listados na tabela 5.42, juntamente com os valores utilizados nos testes, sugeridos pelos autores da técnica.

Tabela 5.42 – Parâmetros usados no método “feasibility pump”

Parâmetro	Descrição	Valor Utilizado
$maxIter_{ST1}$	número máximo de iterações do Estágio 1;	10.000
$maxIter_{ST2}$	número máximo de iterações do Estágio 2;	2.000
$maxStall_{ST1}$	número máximo de iterações sem melhoria do Estágio 1;	70
$maxStall_{ST2}$	número máximo de iterações sem melhoria do Estágio 2;	600
T	número de variáveis que deverão ser invertidas;	10 a 30
lim_frac	limite permitido de não-integralidade, abaixo do qual as variáveis são consideradas inteiras (portanto, não são invertidas)	0,001
α_0	alfa inicial	1,0
δ_α	tolerância para diferenciação entre os pesos α ;	0,5%
φ	fator redutor do peso α ;	0,9
$tempo_max$	tempo máximo de processamento (horas)	6

Os resultados passam a ser mostrados na tabela 5.43.

Tabela 5.43 – Resultados do método “feasibility pump” aplicado ao modelo de fluxo em rede – versão compacta

Cenário	Função Obj [\$]	Tempo [s]	Iterações	Estágio	Bound (\$)	GAP
1	2.408.142,56	2	206	2	219.423,69	90,9%
2	1.099.438,46	1	118	1	320.012,59	70,9%
3	2.436.183,84	2	309	2	120.624,70	95,0%
4	2.643.353,59	3	404	3	302.305,30	88,6%
5	3.415.024,84	2	240	3	977.786,19	71,4%
6	2.994.168,49	2	243	2	182.375,34	93,9%
7	5.261.839,56	6	604	2	397.989,15	92,4%
8	11.922.187,74	10	983	3	623.207,22	94,8%
9	5.329.616,97	6	607	3	235.883,06	95,6%
10	2.909.617,37	4	569	3	110.569,31	96,2%
11	2.982.826,79	4	373	3	1.005.869,64	66,3%
12	3.044.948,75	3	419	2	258.981,11	91,5%
13	10.856.400,76	89	635	3	103.460,71	99,0%
14	8.642.326,27	133	901	3	104.600,50	98,8%
15	9.168.832,08	163	730	3	100.798,86	98,9%
16	6.864.545,30	30	487	3	836.356,58	87,8%
17	9.413.644,22	42	538	3	409.679,92	95,6%
18	6.577.819,55	77	789	3	116.147,01	98,2%
19	17.461.402,32	9.371	715	3	91.124,48	99,5%
20	17.414.360,68	4.544	714	3	79.218,05	99,5%
22	16.454.450,25	420	733	3	92.179,24	99,4%
23	16.650.707,29	301	709	3	132.499,32	99,2%
24	15.293.338,42	523	722	3	112.644,03	99,3%
25	12.781.728,33	14.508	819	3	92.262,69	99,3%
26	8.959.159,11	1.115	561	3	1.585.734,27	82,3%
27	16.517.994,91	1.494	728	3	113.644,97	99,3%
28	19.237.031,33	624	815	3	133.215,49	99,3%
29	13.112.429,42	1.469	743	3	113.395,78	99,1%
30	10.700.178,77	325	536	3	138.425,94	98,7%
42	14.083.802,36	1.673	586	3	124.181,63	99,1%

A análise dos resultados indica que o método “feasibility pump” foi capaz de encontrar uma solução viável em 30 dos 48 problemas (62,5%) em 6 horas de processamento. Embora o método tenha incorporado na função objetivo distância uma parcela contendo a função objetivo do problema original, os resultados indicaram um “gap” médio bastante elevado (93,3%). A solução gerada que apresentou menor “gap” foi o cenário 11 (66,3%). O tempo médio de processamento foi cerca de 20 minutos.

Outra análise importante diz respeito ao estágio em que a solução viável foi gerada. Apenas uma solução foi gerada no estágio 1; no estágio 2, 5 soluções foram geradas; no estágio 3, 24 soluções foram obtidas. Isto indica que as características do modelo tornam difícil o processo de convergência.

5.15 “Feasibility Pump” aplicado ao Modelo de Programação de Frota em Escala de Tempo Discretizada, Sem Deslocamento – Versão II

O método “feasibility pump” também foi processado com o modelo de programação de frota em escala de tempo discretizada, sem deslocamento – versão II. Os mesmos parâmetros e valores que constam na tabela 5.42 foram empregados nos testes com este modelo. Os resultados estão apresentados na tabela 5.44.

A análise dos resultados mostra que em todos os problemas foram encontrados uma solução viável inteira, algo que não aconteceu para os cenários 36 e 46, quando da aplicação do método “branch and cut” ao conjunto de problemas. Assim, desconsiderando estes dois cenários, o “gap” médio foi igual a 92,1%. O tempo médio de processamento dos cenários foi cerca de 3 minutos. Por último, observou-se que 29 problemas foram resolvidos no estágio 1, 19 problemas no estágio 2 e apenas 3 problemas foram resolvidos no estágio 3 do algoritmo “feasibility pump”.

Tabela 5.44 – Resultados do método “feasibility pump” aplicado ao modelo de programação de frota em escala de tempo discretizada, sem deslocamento - II

Cenário	Função Obj [\$]	Tempo [s]	Iterações	Estágio	Bound [\$]	GAP
1	124.089,00	3	52	1	62.367,00	49,7%
2	330.785,00	4	69	1	264.628,00	20,0%
3	572.971,00	3	39	1	0,00	100,0%
4	618.066,00	3	57	1	123.210,00	80,1%
5	2.674.075,00	7	328	2	459.448,00	82,8%
6	22.075,00	5	17	1	8.830,00	60,0%
7	2.988.624,00	6	154	1	170.903,00	94,3%
8	1.409.783,00	6	71	1	282.650,00	80,0%
9	3.754.385,00	8	196	1	155.112,00	95,9%
10	726.434,00	5	54	1	24.170,00	96,7%
11	1.559.524,00	8	194	2	840.612,00	46,1%
12	959.262,00	5	49	1	0,00	100,0%
13	2.882.551,00	27	98	1	0,00	100,0%
14	1.917.203,00	31	134	1	0,00	100,0%
15	4.631.060,00	56	192	2	0,00	100,0%
16	3.220.935,00	112	112	1	1.128.892,25	65,0%
17	1.618.050,00	44	49	1	0,00	100,0%
18	2.269.224,00	72	160	2	0,00	100,0%
19	3.428.416,00	25	54	1	43.961,00	98,7%
20	5.241.753,00	22	58	1	0,00	100,0%
21	1.583.213,00	45	120	1	0,00	100,0%
22	1.910.605,00	35	45	1	15.856,00	99,2%
23	1.612.310,00	78	153	2	0,00	100,0%
24	1.327.472,00	87	53	1	8.048,00	99,4%
25	458.665,00	55	85	1	0,00	100,0%
26	3.968.564,00	103	329	2	847.364,00	78,6%
27	1.488.885,00	43	73	1	0,00	100,0%
28	1.755.605,00	404	557	3	0,00	100,0%
29	542.657,00	77	71	1	0,00	100,0%
30	257.306,00	50	38	1	0,00	100,0%
31	6.884.336,00	680	814	3	0,00	100,0%
32	2.642.348,00	69	117	1	0,00	100,0%
33	6.843.533,00	131	207	2	196.286,00	97,1%
34	6.301.846,00	80	146	1	0,00	100,0%
35	11.315.656,00	512	390	2	176.754,00	98,4%
36	13.863.957,00	531	342	2		
37	11.736.160,00	365	677	3	572.100,00	95,1%
38	6.573.369,00	411	462	2	0,00	100,0%
39	840.575,00	70	101	1	0,00	100,0%
40	10.314.752,00	464	515	2	0,00	100,0%
41	1.873.512,00	302	164	1	0,00	100,0%
42	6.138.971,00	437	272	2	0,00	100,0%
43	2.980.330,00	148	213	1	0,00	100,0%
44	9.291.623,00	202	350	2	0,00	100,0%
45	482.498,00	88	124	1	0,00	100,0%
46	14.003.925,00	261	284	2		
47	15.041.736,00	942	636	2	50.834,00	99,7%
48	6.920.187,00	501	312	2	0,00	100,0%

5.16 Discussão dos Resultados

Nas seções anteriores deste capítulo, foram apresentados resultados da aplicação dos métodos de solução propostos ao problema de programação de frota de apoio a operações “offshore” sujeita à requisição de múltiplas embarcações para uma mesma tarefa. Nesta seção, apresenta-se uma discussão destes resultados.

Os testes foram conduzidos com o propósito de verificar não somente o desempenho dos métodos propostos no capítulo 4, como também a eficácia de conjugar estes métodos de solução. Para isso, utilizou-se uma base de dados, gerada por meio de um procedimento probabilístico que conservasse as propriedades do problema original. A multiplicidade de cenários permitiu avaliar a robustez dos métodos de solução desenvolvidos.

Inicialmente foram testados os métodos “branch and cut” com a formulação matemática de fluxo em rede completa e a compacta. A redução na estrutura do modelo (formulação compacta) influenciou o desempenho do algoritmo “branch and cut”, avaliado pelo número de problemas resolvidos, por um menor valor médio do “gap” e por um tempo menor para resolução dos problemas.

As estimativas dos “gaps” das soluções foram prejudicadas pela dificuldade do pacote computacional CPLEX em estimar os limitantes de solução. A literatura mostra que a versão relaxada de problemas de programação de veículos contendo a estrutura de janela de tempo gera limitantes pobres. Para a formulação compacta de fluxo em rede, “gaps” médios de 73,3%, 57,8% e 84,6% foram obtidos para as classes de 20, 30 e 40 problemas, respectivamente, fazendo com que a avaliação do desempenho das heurísticas e dos métodos combinados (heurística & “branch cut”, heurística & “local branching”, heurística & vns) ficasse igualmente prejudicada.

Observou-se, na aplicação do método “branch and cut”, que para problemas de maior porte é necessário um longo tempo de processamento computacional para encontrar uma primeira solução inteira. Para os problemas de 40 tarefas que tiveram pelo menos uma solução gerada com a versão compacta do modelo de fluxo em rede, este tempo foi, em média, 2 horas e 15 minutos; a heurística construtiva – versão básica, por sua vez, levou, em média, 8 minutos para gerar a primeira solução para os mesmos problemas. Embora esta análise não contemple a qualidade das soluções geradas, é inegável o ganho de desempenho da heurística quando aplicada a problemas de maior porte. Além disso, é importante destacar que, embora os valores médios da função objetivo das heurísticas tenham ficado 13,2% acima dos resultados obtidos pelo método “branch and cut”, o tempo de processamento das heurísticas foi apenas de 6 horas – metade do tempo do método “branch and cut”.

Quatro problemas cujas soluções heurísticas apresentaram altos “gaps” foram processados novamente com as heurísticas de construção, por períodos de 12 horas, com o intuito de verificar se soluções melhores seriam obtidas. Ao invés de cinco processamentos, apenas três foram conduzidos. Os resultados constam na tabela 5.45; as linhas “HC Básica[\$] – 12 horas” e “HC Gulosa[\$] – 12 horas” apresentam os resultados destes testes. Dos quatro cenários avaliados, três tiveram a solução heurística (a melhor entre a versão básica e gulosa) melhorada, embora os novos “gaps” continuem elevados.

Com relação aos cenários 13, 29 e 37, a função objetivo da solução heurística ainda continuou alta em relação à solução obtida pelo método “branch and cut”, sugerindo que outras regras devem ser exploradas e incorporadas nas heurísticas de forma a permitir que soluções ainda melhores sejam geradas. De qualquer maneira, considerando que um dos objetivos era a obtenção de pelo menos uma solução viável para os problemas testados, a estratégia heurística pode ser considerada bem sucedida.

Tabela 5.45 – Heurística construtiva processada por 12 horas

Cenários: Função Objetivo:	13	18	29	37
Melhor Sol. Branch and Cut [\$]	528.829,60	742.715,71	129.730,26	2.400.857,11
HC Básica [\$]	1.347.338,40	1.364.937,21	342.177,36	4.056.352,95
HC Gulosa [\$]	1.168.493,09	1.450.118,93	445.788,38	4.102.716,54
GAP Heurístico	91,1%	91,5%	66,9%	79,4%
Var. HC/BC	121,0%	83,8%	163,8%	69,0%
HC Básica [\$] - 12 horas	1.021.058,31	845.234,06	449.554,74	4.082.719,88
HC Gulosa [\$] - 12 horas	1.214.792,91	1.286.372,79	380.973,36	3.945.391,44
Novo GAP Heurístico	89,9%	86,3%	66,9%	78,8%
Var. HC_12/BC	93,1%	13,8%	163,8%	64,3%

A forma como a heurística foi elaborada, em que as regras para escolha das tarefas e das embarcações são determinadas por sorteio, com uma certa probabilidade, permite avaliar em quais situações que os melhores resultados foram obtidos; isto é, quais probabilidades de aceitação de uma determinada regra que levaram a melhores resultados. Assim, é possível o aprofundamento no conhecimento do problema de programação de embarcações abordado nesta pesquisa, pela experimentação computacional da estratégia heurística e pela avaliação de quais regras que foram mais empregadas durante a construção das melhores soluções heurísticas.

Quanto às heurísticas, é importante destacar que sua aplicabilidade se estende ao problema convencional de programação de veículos com janela de tempo, em que a demanda pode ser atendida por um único veículo – podendo a frota ser homogênea ou não. Para isso, basta especificar que a demanda de cada tarefa é composta por um único veículo.

A aplicação da técnica combinada “heurística construtiva & branch and cut” foi motivada pela expectativa de que o método “branch and cut” conseguiria chegar a soluções melhores, se tivesse uma solução viável inicial - preferencialmente, de alta qualidade – como ponto de partida.

De maneira geral, o sucesso desta estratégia foi razoável, considerando que a redução média observada na função objetivo após as 12 horas (6 horas de processamento da heurística e 6 horas de processamento do “branch and cut”) foi de 13,4%. A comparação do desempenho deve levar em conta o fato de que o método “branch and cut” atuou apenas por um período de 6 horas.

Se a comparação fosse entre a aplicação do método “branch and cut” com e sem solução inicial, por um mesmo período, seria esperado um melhor desempenho para os casos onde há solução inicial, a menos que esta esteja contida em uma região de ótimo local. Com o intuito de checar o efeito da utilização de solução inicial no desempenho do “branch and cut”, foram escolhidos os cenários 13, 22 e 27 para serem processados por um período de 6 horas com o algoritmo “branch and cut” em duas situações: com solução inicial heurística e sem solução inicial. As figuras 5.12, 5.13 e 5.14 mostram a evolução da função objetivo em ambas as situações, para cada cenário.

No cenário, 13, embora o valor da função objetivo da solução inicial heurística seja aproximadamente metade do valor da função objetivo da solução inicial gerada pelo “branch and cut”, esta solução está contida em uma região de ótimo local e o “branch and cut” não conseguiu sair desta vizinhança. No cenário 22, o valor da função objetivo da solução inicial heurística foi aproximadamente 10% do valor da função objetivo da solução gerada pelo “branch and cut”. Embora ambas as soluções tivessem sido melhoradas, a solução heurística ficou em um patamar que não foi alcançado pela estratégia exata.

Figura 5.12 – Comparação de desempenho do “branch and cut” aplicado ao cenário #13 com e sem solução inicial

Figura 5.13 – Comparação de desempenho do “branch and cut” aplicado ao cenário #22 com e sem solução inicial

No cenário 27, o valor da função objetivo da solução inicial gerada pelo “branch and cut” foi quase 8 vezes maior que o valor da função objetivo da solução inicial heurística. Mesmo assim, a melhor solução após as 6 horas, corresponde ao caso em que não houve solução inicial.

Figura 5.14 – Comparação de desempenho do “branch and cut” aplicado ao cenário #27 com e sem solução inicial

Os testes feitos com o método combinado “heurística construtiva & variable neighborhood search” visaram contornar a limitação constatada no “branch and cut” na exploração do espaço de solução do problema, comprovadas pelos gráficos das figuras 5.12 a 5.14. O refinamento (ou melhoria) de uma solução de um problema contendo centenas ou milhares de variáveis de decisão binárias pode ser lento, caso soluções melhores estejam distantes da solução corrente, elevando excessivamente o tempo para a exploração da vizinhança.

Assim, concentrar a busca em uma região menor do espaço de solução é uma estratégia potencialmente promissora, desde que a definição desta região e do tempo a ser empregado na busca seja adequada para o problema. Esta adequação pode ser conseguida pela calibração dos parâmetros que definem a estrutura da vizinhança e é específica a cada problema. Além disso, o método deve ser capaz de identificar regiões de ótimo local e agir de maneira diversificativa.

Os resultados obtidos pela estratégia combinada “heurística & vns” (função objetivo 22,6% menor que o “branch and cut” isoladamente) foram melhores que os obtidos pela “heurística construtiva & branch and cut”, sendo este desempenho atribuído à forma como a vizinhança é explorada. Cabe ressaltar que a calibração não foi exaustiva, podendo outras estruturas de vizinhança se mostrarem mais eficazes.

O método de busca em vizinhança “local branching” foi o que apresentou melhores resultados para os cenários testados nesta pesquisa. Conceitualmente, as técnicas “local branching” e “vns” se assemelham quanto à estratégia de particionar ou subdividir o espaço de solução, concentrando a busca em regiões menores. A técnica “local branching”, contudo, mostrou-se mais eficaz no processo de construção, exploração da vizinhança e diversificação. Um indicador que comprova este sucesso é a variabilidade dos resultados, medido pelo coeficiente de variação dos resultados, para as várias estruturas de vizinhança de cada um dos métodos: 12,0% na técnica “local branching” contra 6,5% da técnica “vns”.

Com o intuito de verificar o desempenho da técnica “local branching” sem o uso de solução inicial, foi feito um teste com os 48 cenários, utilizando para cada um deles o modelo de fluxo em rede – versão compacta, e a estrutura de vizinhança que gerou melhores resultados (quando processado a partir das soluções iniciais heurísticas). É importante lembrar que, na ausência de solução inicial, a técnica “local branching” recorre ao método “branch and cut” até que uma solução viável seja

gerada; só então que o processo de busca em vizinhança é iniciado. Para estes testes, a função objetivo ficou, em média, 14,6% acima dos valores obtidos pela técnica combinada “heurística construtiva & local branching”. Apesar de ser uma diferença pequena, é importante destacar que em 10 cenários o “local branching” não conseguiu sequer gerar uma solução inicial.

Em função da dificuldade em solucionar os problemas de maior porte, foi testado, em uma fase posterior da pesquisa, o método “feasibility pump”, com a expectativa de que em todos os problemas seria possível obter pelo menos uma solução viável. Isto não se confirmou para os modelos de fluxo em rede, sendo que em apenas 30 dos 48 problemas foram geradas soluções viáveis. Quanto aos cenários de maior porte, foram geradas soluções para 6 problemas de 30 tarefas, e em apenas 1 problema contendo 40 tarefas. Quanto aos modelos de programação de frota em escala de tempo discretizada, sem deslocamento, o método “feasibility pump” teve sucesso na construção das soluções para todos os cenários, sendo estas, contudo, de baixa qualidade.

É importante destacar que também foi testada a heurística “relax-and-fix”, apresentada na seção 2.4.4. Contudo, por não haver um critério robusto que pudesse ser aplicado na formação dos conjuntos de variáveis binárias, não se obteve sucesso com a uso desta estratégia.

A tabela 5.46 agrupa os resultados obtidos com a aplicação de todos os métodos, lembrando que os métodos combinados utilizaram o modelo de fluxo em rede – versão compacta.

Tabela 5.46 – Tabela Resumo – Valor da função objetivo da melhor solução gerada pelos diversos métodos utilizados

Cenário	Branch and Cut Fluxo Rede Completo [\$]	Branch and Cut Fluxo Rede Compacto [\$]	Heurística Construtiva - Versão Básica [\$]	Heurística Construtiva - Versão Gulosa [\$]	Heurística & Branch and Cut [\$]	Heurística & VNS [\$]	Heurística & LB [\$]	Feasibility Pump [\$]	Melhor Bound [\$]
1	221.478,92	221.478,92	222.232,11	222.500,82	221.518,08	221.478,92	221.478,92	2.408.142,56	219.423,69
2	320.903,43	320.918,10	321.915,66	321.637,70	321.559,99	320.903,43	320.903,43	1.099.438,46	320.012,59
3	121.749,50	121.752,49	124.116,36	122.852,10	121.749,50	121.749,50	121.749,50	2.436.183,84	120.624,70
4	305.256,69	304.837,97	306.052,21	306.119,74	304.807,98	304.807,98	304.807,98	2.643.353,59	302.305,30
5	979.990,95	980.589,10	984.922,71	984.763,82	981.220,10	979.632,35	979.632,35	3.415.024,84	977.786,19
6	182.949,79	182.930,96	185.772,35	186.075,75	183.097,48	182.898,78	182.898,78	2.994.168,49	182.375,34
7	402.002,50	401.906,02	401.329,05	401.898,86	401.906,02	401.906,02	401.906,02	5.261.839,56	397.989,15
8	629.357,48	629.357,48	731.154,33	634.070,89	629.357,48	629.357,48	629.357,48	11.922.187,74	623.207,22
9	237.501,42	238.231,94	238.584,22	238.543,15	237.171,66	237.141,21	237.141,21	5.329.616,97	235.883,06
10	208.245,72	207.867,45	210.159,01	210.148,98	207.886,55	207.867,45	207.867,45	2.909.617,37	110.569,31
11	1.009.664,51	1.012.575,67	1.012.864,75	1.012.640,96	1.010.878,12	1.009.457,92	1.009.457,92	2.982.826,79	1.005.869,64
12	261.365,23	261.365,23	277.566,97	286.419,57	261.365,23	261.365,23	261.365,23	3.044.948,75	258.981,11
13	528.829,60	629.656,66	1.347.338,40	1.168.493,09	757.403,54	697.342,94	272.058,32	10.856.400,76	103.460,71
14	1.043.208,27	909.638,55	776.064,40	802.843,52	679.682,80	331.196,05	250.156,93	8.642.326,27	104.600,50
15	151.159,12	107.582,21	328.117,70	221.739,00	107.588,11	107.623,00	107.576,17	9.168.832,08	100.798,86
16	1.986.652,01	1.827.501,26	1.972.111,00	1.966.349,49	1.871.660,69	1.779.165,09	1.778.281,79	6.864.545,30	836.356,58
17	1.575.170,98	556.027,95	1.581.083,43	1.342.055,80	643.507,02	623.335,08	552.317,59	9.413.644,22	409.679,92
18	1.258.439,29	742.715,71	1.364.937,21	1.450.118,93	1.146.254,86	474.254,59	581.159,21	6.577.819,55	116.147,01
19	715.186,72	1.203.817,12	614.186,52	654.700,47	484.798,70	503.647,28	504.489,25	17.461.402,32	91.124,48
20	2.457.835,57	1.468.270,15	465.091,18	505.794,93	104.039,81	183.381,14	101.841,44	17.414.360,68	79.218,05
21	5.578.286,28	4.107.550,39	568.152,02	544.966,86	476.047,77	201.938,15	197.284,65		83.966,34
22	1.361.556,70	1.662.005,81	464.351,96	421.103,40	156.436,33	141.982,05	141.969,62	16.454.450,25	92.179,24
23	1.643.079,91	444.872,75	688.242,62	653.564,80	504.085,24	231.428,39	217.024,67	16.650.707,29	132.499,32
24	1.029.994,56	807.196,86	743.492,91	814.668,88	709.262,12	324.787,96	302.190,76	15.293.338,42	112.644,03

Tabela 5.46 – Tabela Resumo – Valor da função objetivo da melhor solução gerada pelos diversos métodos utilizados

Cenário	Branch and Cut Fluxo Rede Completo [\$]	Branch and Cut Fluxo Rede Compacto [\$]	Heurística Construtiva - Versão Básica [\$]	Heurística Construtiva - Versão Gulosa [\$]	Heurística & Branch and Cut [\$]	Heurística & VNS [\$]	Heurística & LB [\$]	Feasibility Pump [\$]	Melhor Bound [\$]
25		1.891.799,93	1.091.615,51	894.144,48	860.079,71	453.276,14	446.849,82	12.781.728,33	92.262,69
26	1.762.501,93	1.660.976,06	2.258.509,72	2.101.298,45	1.658.751,52	1.673.792,91	1.665.117,87	8.959.159,11	1.585.734,27
27	1.570.679,79	162.732,27	261.781,53	314.518,48	137.494,64	171.219,00	120.890,24	16.517.994,91	113.644,97
28	272.703,36	863.941,85	536.491,34	427.524,64	350.545,94	372.608,98	229.253,05	19.237.031,33	133.215,49
29	1.003.693,14	129.730,26	342.177,36	445.788,38	341.356,12	170.820,56	129.554,30	13.112.429,42	113.395,78
30	1.297.483,36	142.818,37	179.769,96	345.171,37	140.242,16	148.521,03	140.621,86	10.700.178,77	138.425,94
31	3.425.403,80	5.589.106,09	1.519.754,15	1.275.137,61	1.275.853,13	373.357,29	321.455,87		83.382,64
32			3.242.852,79	3.170.537,77	3.170.719,32	3.167.501,10	2.393.678,30		79.391,98
33			2.976.259,76	3.375.938,17	2.977.925,52	2.426.664,07	1.142.210,20		96.186,69
34	4.309.820,86	2.007.944,88	1.068.807,63	1.255.833,88	1.066.269,82	502.645,93	144.197,43		106.465,54
35		7.388.233,59	5.430.102,71	6.180.223,98	5.429.430,28	4.931.665,59	3.273.185,73		203.516,78
36			3.448.200,26	4.060.579,14	3.452.382,83	3.092.796,47	2.520.446,28		115.302,92
37		2.400.857,11	4.056.352,95	4.102.716,54	3.509.911,72	3.778.832,89	2.850.312,84		837.441,35
38			1.421.661,02	1.277.541,12	1.277.190,45	1.278.366,20	764.206,00		119.359,60
39	11.871.173,16	5.245.226,34	909.450,35	784.203,96	785.029,32	785.029,32	205.926,98		137.477,49
40		2.572.777,40	3.753.034,33	2.704.305,47	1.701.359,01	2.324.458,21	677.195,37		344.488,74
41			3.190.959,85	3.706.154,91	3.132.253,44	1.897.420,78	590.864,31		133.939,73
42	1.439.097,26	598.289,07	486.565,66	575.927,65	268.552,51	487.333,19	131.421,16	14.083.802,36	124.181,63
43	10.005.453,44		990.130,59	1.330.596,19	975.165,33	985.665,05	535.446,39		104.870,47
44			1.858.519,96	1.209.300,50	1.202.736,65	1.006.025,40	740.323,95		115.798,42
45		1.763.845,32	585.961,92	492.293,54	417.676,59	492.175,03	389.711,10		102.533,83
46			1.999.227,87	2.942.312,64	1.961.319,92	2.003.900,94	1.067.374,82		118.747,22
47			3.896.796,23	3.702.733,21	3.350.962,08	3.523.426,07	2.512.233,89		196.218,86
48			2.665.528,05	2.167.414,40	2.160.298,14	1.481.311,01	1.740.538,06		124.117,00

Estes resultados, comparados entre si, confirmam o melhor desempenho da estratégia “heurística construtiva & local branching”. A tabela 5.47 apresenta o desempenho dos vários métodos empregados, tendo um deles como solução de referência. Por exemplo, todos os outros métodos foram melhores do que os resultados obtidos pelo “branch and cut” aplicado à formulação completa de fluxo em rede. O método “feasibility pump” foi excluído desta análise por ter conseguido resolver um número limitado de problemas e por ter gerado soluções de baixa qualidade.

Tabela 5.47 – Comparação entre os métodos de solução quanto ao valor da função objetivo da melhor solução gerada

Solução Referência:	Método de Solução						
	BC Flx Completo	BC Flx Compacto	HC Básica	HC Gulosa	HC & BC	HC & VNS	HC & LB
BC Flx Completo		-11,09%	-15,99%	-19,85%	-33,22%	-38,75%	-44,50%
BC Flx Compacto	99,08%		10,18%	10,56%	-15,43%	-24,76%	-35,10%
HC Básica	163,75%	50,70%		1,59%	-17,75%	-26,69%	-41,22%
HC Gulosa	147,91%	57,72%	1,99%		-17,78%	-26,75%	-40,98%
HC & BC	292,36%	128,24%	36,83%	36,02%		-7,88%	-28,17%
HC & VNS	363,14%	191,71%	61,31%	59,65%	24,66%		-19,80%
HC & LB	665,00%	324,86%	132,92%	133,27%	78,98%	47,54%	
# Problemas Resolvidos	34	38	48	48	48	48	48

A comparação contemplando a melhor solução exata (melhor solução “branch and cut” aplicada aos modelos completo e compacto), a melhor solução heurística (entre a versão básica e gulosa) e as melhores soluções “vns” e “local branching” estão resumidas na tabela 5.48. O número de problemas de referência do método “branch and cut”, usados para estes cálculos aumentou de 38 para 39, dado que um cenário tinha sido resolvido apenas com o modelo completo.

A análise da tabela 5.48 remete à questão de como utilizar os métodos desenvolvidos para resolução de um problema de programação de frota. A melhor estratégia para resolver este problema, com base nos métodos desenvolvidos, é recorrer às heurísticas para obtenção das soluções iniciais. O número de processamentos, bem como a duração destas dependerá da disponibilidade de recursos computacionais. A dispersão observada nos resultados dos testes das heurísticas, de aproximadamente 13%, sugere que o maior número possível de testes, tanto com a versão básica, quanto com a versão gulosa, poderá colaborar para que soluções de boa qualidade sejam geradas. A melhor solução heurística obtida deverá ser submetida à técnica “local branching” que, por sua vez, requer que diferentes tipos de vizinhança sejam testados.

Tabela 5.48 – Comparação entre os métodos de solução

Solução Referência	Método de Solução				
	Branch and Cut	Heurística	HC & BC	HC & VNS	HC & LB
Branch and Cut		5,43%	-13,36%	-22,60%	-34,04%
Heurística	72,18%		-13,00%	-22,49%	-38,17%
HC & BC	132,15%	26,52%		-7,88%	-28,17%
HC & VNS	183,02%	49,20%	24,66%		-19,80%
HC & LB	318,40%	115,29%	78,98%	47,54%	
# Problemas Resolvidos	39	48	48	48	48

Assim, fica claro que o processo de programação da frota na operação de reboque e lançamento de linhas de ancoragem demanda considerável esforço computacional com os métodos de solução desenvolvidos. Isto é coerente com o tamanho do espaço de solução dos problemas que estão sendo resolvidos, embora seja indesejável, do ponto de vista prático. O capítulo de conclusão desta pesquisa cita algumas técnicas que poderão ser empregadas para melhorar o desempenho do método “branch and cut” em problemas de grande porte.

Apesar do elevado tempo de processamento, é importante lembrar que o processo de decisão, na prática, ocorre com intervalo suficientemente grande para que novas soluções sejam geradas.

Os modelos desenvolvidos nesta pesquisa enfatizam o aspecto da condição inicial das embarcações. Em um determinado instante, quando se inicia um novo horizonte de planejamento, várias embarcações poderão estar em operação, alocadas a diversas tarefas. Os seus instantes e locais de disponibilidade inicial (para o novo horizonte de planejamento) constituem parâmetros de entrada dos modelos, e refletem o término destas campanhas. Assim, caso surjam, em qualquer momento, novas tarefas que obriguem a redefinição do plano de alocação da frota, a reprogramação desta é possível de ser feita, e necessariamente respeitará as embarcações que estiverem em operação, bastando informar, de maneira adequada, a condição inicial das embarcações.

Quanto aos modelos de programação de frota em escala de tempo discretizada, a ausência dos tempos de viagem nas versões “sem deslocamento - I e II” não compromete necessariamente a aplicação de seus resultados, quando estes são usados com o intuito de definir a macro-programação das tarefas. Entende-se por isso o processo de definição e ajuste dos instantes desejados de início de cada tarefa conjugado com uma avaliação preliminar da disponibilidade de recursos para execução das mesmas. A duração de cada tarefa, que por hipótese é conhecida e determinística, é estipulada com base na experiência dos programadores em relação às condições de mar e à complexidade da operação, que são os fatores mais importantes para estimar a duração das tarefas. Para efeito de uma macro-programação, pode-se estipular a duração das tarefas com uma folga isto é, com uma margem de segurança que contemplaria o tempo para a frota se posicionar, baseando-se em distâncias médias percorridas pelas embarcações.

Por último, o desempenho do método “branch and cut” aplicado ao modelo de programação de frota em uma escala de tempo discretizada, contemplando o deslocamento das embarcações, se mostrou limitado a problemas de pequeno porte. Isto se deve à discretização da escala de tempo que interfere diretamente no número de variáveis de decisão.

6 CONCLUSÕES E RECOMENDAÇÕES

Este capítulo apresenta as principais conclusões e recomendações a respeito da pesquisa focada na PROGRAMAÇÃO DE FROTA DE APOIO A OPERAÇÕES “OFFSHORE” SUJEITA À REQUISIÇÃO DE MÚLTIPLAS EMBARCAÇÕES PARA UMA MESMA TAREFA.

Foi estudado o problema de programação de embarcações AHTS usadas no reboque de plataformas de perfuração e de produção de petróleo, e no manuseio de suas âncoras. Enquanto que as plataformas de perfuração são continuamente realocadas na preparação de novos poços para a etapa produtiva, as plataformas de produção ficam dedicadas a determinadas áreas, que englobam um conjunto de poços, por vários anos. O reboque destas unidades marítimas pode exigir a participação de mais de uma embarcação AHTS para atender às exigências de segurança e desempenho.

O emprego de embarcações AHTS no processo de lançamento das linhas de ancoragem consiste em uma operação coordenada de múltiplas embarcações. Esta exigência se dá pela complexidade da operação em si, acentuada pela alta lâmina d’água das regiões exploradas e pelas condições de mar.

Assim, a junção das características que este problema apresenta faz dele uma variante mais complexa do problema clássico de programação de veículos.

O presente tema de pesquisa, apesar de ser motivado por uma operação vivenciada pela PETROBRAS, reflete a problemática enfrentada por outras empresas atuando nas diversas frentes de exploração “offshore” no mundo, a de alocar,

simultaneamente, múltiplas embarcações de uma frota dedicada de AHTS a tarefas de reboque e lançamento de linhas de ancoragem.

A revisão da literatura foi focada na área de programação de veículos e em modelos de planejamento de “frota industrial”. As aplicações descritas na bibliografia e apresentadas no capítulo 2, são voltadas para casos em que o atendimento de cada tarefa é feito por um único veículo, não contemplando, portanto, o emprego de múltiplos veículos para uma mesma tarefa, que é uma característica fundamental do problema tratado na presente pesquisa. Assim, fez-se necessário elaborar novos modelos matemáticos, bem como novos métodos de solução, específicos para o problema desta pesquisa, que fossem capazes de lidar e incorporar todos os aspectos descritos nos capítulos 1 e 3.

Duas estratégias distintas de modelagem foram seguidas. Na primeira, optou-se pela representação do fluxo de embarcações em uma rede, cujos nós indicam as tarefas a serem executadas, e os arcos, as possíveis viagens das embarcações. Associado aos nós da rede, tem-se um conjunto de parâmetros relativos a cada tarefa, entre eles as janelas de tempo, o instante de início desejado, a penalidade por atraso e a quantidade de embarcações por classe. Os arcos da rede indicam os possíveis fluxos de embarcações e impedem que viagens inconsistentes sejam geradas como, por exemplo, uma embarcação sair de um determinado nó e atender uma outra tarefa cuja janela de tempo já tenha expirado.

Três modelos foram desenvolvidos com a estratégia de representação de fluxo em rede. No primeiro, denominado de “versão completa”, todos os elementos essenciais do problema foram incorporados, havendo, contudo, um conjunto de restrições redundantes. Estas foram suprimidas, gerando a “versão compacta” de fluxo em rede. Além destes dois modelos, foi elaborada uma versão em que todos os possíveis conjuntos de embarcações, capazes de atender a uma tarefa, eram

previamente gerados e a escolha do conjunto ficava por conta do modelo matemático. Esta versão foi utilizada apenas em estágios iniciais da pesquisa, uma vez que o número total de conjuntos introduziu uma sobrecarga de variáveis de decisão (variáveis binárias), limitando a utilização de métodos exatos de solução.

A segunda estratégia focou a determinação do instante de início de cada tarefa conjuntamente com a escolha das embarcações para realizar a tarefa, utilizando-se uma escala de tempo discretizada. O modelo define os instantes de início das tarefas respeitando as restrições operacionais do problema e impedindo que uma mesma embarcação seja alocada a mais de uma tarefa ao mesmo tempo. Em dois dos modelos propostos, optou-se por desconsiderar os tempos de viagem das embarcações, uma vez que a ordem de grandeza dos custos variáveis de seus deslocamentos é inferior à ordem de grandeza das penalidades por atraso. Isto torna mais fácil o processo de busca da solução, já que o número de variáveis de decisão é menor, limitando, contudo, a aplicabilidade destes modelos. Esta limitação pode ser, em parte, contornada, ao se incorporar na duração das tarefas um tempo adicional correspondente aos tempos médios de deslocamento. O terceiro modelo, desenvolvido segundo a estratégia de determinação dos instantes de início das tarefas, incorporou as variáveis de fluxo das embarcações, permitindo contemplar os tempos de viagem. Contudo, a quantidade total de variáveis binárias tornou proibitiva a aplicação de métodos exatos, devido a restrições de capacidade de processamento computacional.

Os métodos empregados para solucionar o problema seguiram 2 estratégias distintas, e que posteriormente foram integradas: 1) a abordagem heurística que, de maneira probabilística, constrói soluções por meio da aplicação das regras de decisão escolhidas (ou, sorteadas) de acordo com o “status” do problema, sendo este expresso pela criticidade da janela de tempo das tarefas em fila; 2) a abordagem otimizante, que consistiu na aplicação do algoritmo exato “branch and bound”,

complementado pela adição de desigualdades válidas aos problemas relaxados em cada nó da árvore. Ambas as estratégias constituem formas consagradas de solução de problemas combinatórios, particularmente, de problemas de programação de veículos.

A integração dos métodos se fez necessária pela incapacidade do método “branch and cut” gerar soluções para todos os problemas testados, muito embora houvesse pelo menos uma solução viável para todos os problemas. Neste processo, utilizou-se como solução inicial para o método “branch and cut” a melhor solução heurística, isto é, a de menor valor da função objetivo. Alguns testes mostraram que não existe uma relação entre a qualidade da solução inicial com a qualidade da solução final.

Nos métodos de busca “local branching” e “variable neighborhood search”, a vizinhança é definida pela adição de restrições que delimitam subregiões do espaço de solução, e a busca se dá de maneira enumerativa, implicitamente, por meio do algoritmo “branch and bound”. Estas estratégias se mostraram eficazes e geraram os melhores resultados desta pesquisa. É importante lembrar, contudo, que a eficácia destes métodos foi devida à existência da solução inicial heurística.

Do ponto de vista de aplicabilidade dos métodos desenvolvidos a outros problemas de programação de veículos, pode-se dizer que as heurísticas construtivas possuem uma estrutura facilmente adaptável ao problema clássico de roteirização sujeito a janelas de tempo, bastando adaptar os parâmetros de entrada, informando que apenas um veículo é necessário para atender à demanda. Não há, contudo, garantia de que elas serão competitivas com os métodos existentes, uma vez que foram elaboradas explorando características específicas do problema desta pesquisa.

Já os métodos “local branching” e “variable neighborhood search” são totalmente ajustáveis a outras formulações matemáticas (ou, a outros problemas), uma vez que a forma de construção da vizinhança independe da estrutura particular de um problema.

A pesquisa desenvolvida se mostrou limitada quanto à capacidade de resolver problemas de grande porte. Mesmo conjugando os diversos métodos de solução, os “gaps” observados são elevados. Além disso, o método “feasibility pump”, implementado especificamente com a finalidade de gerar soluções iniciais, não foi capaz de resolver os problemas de maior porte.

Alguns possíveis desdobramentos e linhas de pesquisa que poderiam ser desenvolvidas, a partir dos resultados obtidos, são:

- Avaliação da introdução de desigualdades válidas eficazes para os problemas de roteirização de veículos com janela de tempo – os resultados já conhecidos da estrutura poliédrica de problemas de roteirização e programação de veículos (Bard et al., 2002; Kohl et al., 1999) poderão ser explicitamente incorporados à formulação matemática ou aos problemas de separação resolvidos durante o “branch and cut”;
- Formulação matemática alternativa – além da formulação de fluxo em rede e da formulação de programação de frota em uma escala de tempo discretizada, o mesmo problema poderia ser modelado tendo como decisões principais a seqüência de atendimento das tarefas e a escolha dos conjuntos de embarcações, sendo os instantes de início das tarefas, consequência das decisões tomadas;
- Paralelização dos algoritmos de solução – a disponibilidade de infra-estrutura computacional permite que uma estrutura distribuída dos algoritmos desenvolvidos seja elaborada. Em particular, para o método VNS, García-López et al. (2002) comprovam a eficácia da estratégia de paralelização;

- Relaxação lagrangeana – tendo em vista as dificuldades de obtenção de limitantes de boa qualidade para os problemas de roteirização com janela de tempo, seria importante avaliar a eficácia de relaxar as restrições críticas deste problema segundo a técnica da relaxação lagrangeana, que tem se mostrado uma alternativa eficiente na resolução desta classe de problemas (Kallehauge et. al, 2006);
- Aperfeiçoamento da heurística construtiva – a heurística poderá incorporar outras regras de decisão que sejam mais eficazes, de forma a gerar resultados ainda melhores. Sugere-se comparar as decisões tomadas pela heurística com as decisões que o método “branch and cut” realizou e verificar se é possível abstrair novas regras.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ACHTERBERG, T.; BERTHOLD, T. Improving the feasibility pump. Konrad-Zuse-Zentrum für Informationstechnik Berlin. ZIB-Report 05-42, 2005.

APPELGREN, L.H. A column generation algorithm for a ship scheduling problem. **Transportation Science**, 3, p.53-68, 1969.

APPELGREN, L.H. Integer programming methods for a vessel scheduling problem. **Transportation Science**, 5, p.64-78, 1971.

AUGUSTO, O.B.; ANDRADE, B.L.R.; ROSSI, R.R. Otimização de recursos para a operação de instalação de âncoras de equipamentos *offshore*. **Pesquisa Operacional**, 22:3, p.305-322, 2002.

BALAS, E.; SCHMIETA, S.; WALLACE, C. Pivot and shift – a mixed integer programming heuristic. **Discrete Optimization**, 1, p.3-12, 2004.

BARD, J.F.; KONTORAVDIS, G.; YU, G. A branch-and-cut procedure for the vehicle routing problem with time windows. **Transportation Science**, 36:2, p.250-269, 2002.

BAUSCH, D.O.; BROWN, G.G.; RONEN, D. Scheduling short-term marine transport of bulk products. **Maritime Policy & Management**, 25:4, p.335-348, 1998.

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

BRÄYSY, O.; GENDREAU, M. Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. **Transportation Science**, 39:1, p.104-118, 2005a.

BRÄYSY, O.; GENDREAU, M. Vehicle routing problem with time windows, Part II: Metaheuristics. **Transportation Science**, 39:1, p.119-139, 2005b.

BREMER, W.M.; PERAKIS, A.N. An operational tanker scheduling optimization system: model implementation, results and possible extensions. **Maritime Policy & Management**, 19:3, p.189-199, 1992.

BRISKIN, L. Selecting delivery dates in the tanker scheduling problem. **Transportation Science**, 12:6, p.B-224-B-235, 1966.

BRØNMO, G.; CHRISTIANSEN, M.; FAGERHOLT, K.; NYGREEN, B. A multi-start local search heuristic for ship scheduling – a computational study. **Computers & Operations Research**, 2005.

BROWN, G.G.; GRAVES, G.W.; RONEN, D. Scheduling ocean transportation of crude oil. **Management Science**, 33:3, p.335-346, 1987.

BRUCKER, P.; DREXL, A.; MÖHRING, R.; NEUMANN, K.; PESCH, E. Resource-constrained project scheduling: notation, classification, models and methods. **European Journal of Operational Research**, 112, p.3-41, 1999.

CHO, S.C.; PERAKIS, A.N. Optimal liner fleet routing strategies. **Maritime Policy & Management**, 23:3, p.249-259, 1996.

CHO, S.C.; PERAKIS, A.N.; An improved formulation for bulk cargo ship scheduling with a single loading port. **Maritime Policy & Management**, 28:4, p.339-345, 2001.

CHRISTIANSEN, M.; FAGERHOLT, K. Robust ship scheduling with multiple time windows. **Naval Research Logistics**, 49, p.611-625, 2002.

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

CLARKE, G.; WRIGHT, J.W. Scheduling of vehicles from a central depot to a number of delivery points. **Operations Research**, 12, p.568-581, 1964.

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

CORDEAU, J.F., TOTH P., VIGO D. A survey of optimization models for train routing and scheduling. **Transportation Science**, 32, p.988-1005, 1998.

CPLEX: ILOG CPLEX 10.0. User's manual and reference guide, 2006.

DANNA E.; ROTHBERG, E.; LE PAPE, C. Exploring relaxation induced neighborhoods to improve MIP solutions. **Mathematical Programming, Ser. A**, 102, p.71-90, 2005.

DESROCHERS, M.; LENSTRA, J.K.; SAVELSBERGH, M.W.P. A classification scheme for vehicle routing and scheduling problems. **European Journal of Operational Research**, 46, p.322-332, 1990.

DESROCHERS, M.; JONES, C.V.; LENSTRA, J.K.; SAVELSBERGH, M.W.P.; STOUGIE, L. Towards a model and algorithm management system for vehicle routing and scheduling problems. **Decision Support Systems**, 25, p.109-133, 1999.

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, 1995. p. 35–139.

DYKE, K.V. **Fundamentals of Petroleum**. PETEX, 1997. 370p.

FAGERHOLT K. A computer-based decision support system for vessel fleet scheduling – Experience and future research. **Decision Support Systems**, 37, p.35-47, 2004.

FAGERHOLT K., HEIMDAL S.I., LOKTU A. Shortest path in the presence of obstacles: an application to ocean shipping. **Journal of the Operational Research Society**, 51, p.683-688, 2000.

FAGERHOLT, K. Evaluating the trade-off between the level of customer service and transportation costs in a ship scheduling problem. **Maritime Policy & Management**, 27:2, p.145-153, 2000.

FAGERHOLT, K. Optimal fleet design in a ship routing problem. **International Transactions in Operational Research**, 6, p.453-464, 1999.

FAGERHOLT, K.; CHRISTIANSEN, M. A combined ship scheduling and allocation problem. **Journal of the Operational Research Society**, 51, p.834-842, 2000.

FAGERHOLT, K.; LINDSTAD, H. Optimal policies for maintaining a supply service in the Norwegian Sea. **OMEGA**, 28, p.269-275, 2000.

FISCHETTI, M.; GLOVER, F.; LODI, A. The feasibility pump. **Mathematical Programming, Ser. A**, 104, p. 91-104, 2005.

FISCHETTI, M.; LODI, A. Local branching. **Mathematical Programming, Ser. B**, 98, p. 23-47, 2003.

FISHER, M.L.; ROSENWEIN, M.B. An interactive optimization system for a bulk-cargo ship scheduling. **Naval Research Logistics**, 36, p.27-42, 1989.

FRANCIS, P.; ZHANG, G.; SMILOWITZ, K. Improved modeling and solution methods for the multi-resource routing problem. **European Journal of Operational Research**, 180, p.1045-1059, 2007.

GARCÍA-LÓPEZ F.; MELIÁN-BATISTA, B.; MORENO-PÉREZ, J.A.; MORENO-VEGA, J.M. The parallel variable neighborhood search for the p -median problem. **Journal of Heuristics**, 8, p.375-388, 2002.

GENDREAU, M.; HERTZ, A.; LAPORTE, G. A new insertion and postoptimization procedures for the traveling salesman problem. **Operations Research**, 40, p.1086-1093, 1992.

GLOVER, F.; LAGUNA, M. **Tabu Search**. Massachusetts: Kluwer Academic Publishers, 1999.

GOLDEN, B.; ASSAD, A.; LEVY, L.; GHEYSENS, E. The fleet size and mix vehicle routing problem. **Computers & Operations Research**, 11, p.49-66, 1984.

HANSEN, P.; MLADENović, N. Variable neighborhood search: principles and applications. **European Journal of Operational Research**, 130, p.449-467, 2001.

HANSEN, P.; MLADENović, N. A tutorial on variable neighborhood search. *Les Cahiers du GERAD*, G-2003-46, 2003.

HANSEN, P.; MLADENović, N.; UROŠEVIĆ, D. Variable neighborhood search and local branching. **Computers & Operations Research**, 33, p.3034-3045, 2006.

HOLLAND, J.H. **Adaptation in Natural and Artificial Systems**. Ann Arbor, Michigan: University of Michigan Press, 1975.

JARAMILLO, D.I.; PERAKIS, A.N. Fleet deployment optimization for liner shipping Part 2. Implementation and results. **Maritime Policy & Management**, 18:4, p.235-262, 1991.

KALLEHAUGE, B.; LARSEN, J.; MADSEN, O.B.G. Lagrangian duality applied to the vehicle routing problem with time windows. **Computers and Operations Research**, 33:5, p.1464-1487, 2006.

KIM, S.H.; LEE, K.K. An optimization-based decision support system for ship scheduling. **Computers Industrial Engineering**, 33:3-4, p.682-692, 1997.

KIRKPATRICK, S.; GELATT, C.D.; VECCHI, M.P. Optimization by simulated annealing. **Science**, 220, p.671-680, 1993.

KOHL, N.; DESROSIERS, J.; MADSEN, O.B.G.; SOLOMON, M.M.; SOUMIS, F. 2-Path cuts for the vehicle routing problem with time windows. **Transportation Science**, 33:1, p.101-116, 1999.

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

LEI, L.; FAN, C.; BOILE, M.; THEOFANIS, S. Collaborative vs. non-collaborative container-vessel scheduling. **Transportation Research Part E**, no prelo, 2007.

LIN, S. Computer solutions of the traveling salesman problem. **Bell Systems Technology Journal**, 44, p.2245-2269, 1965.

LIU, J. **Solving real-life transportation scheduling problems**. 2003. 173p. Tese (Doutorado) – Industrial and Systems Engineering, University of Florida, 2003.

MANNE, A.S. On the job-shop scheduling problem. **Operations Research**, 8:2, p.219-223, 1960.

MLADENović, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, 24, p.1097-1100, 1997.

MORTON, T.E.; PENTICO, D.W. **Heuristic scheduling systems – with applications to production systems and project management**. New York: JOHN WILEY & SONS, INC., 1993. 695p.

NAGLE, F.J.M. **Organização e gestão da inovação tecnológica para produção “offshore” de petróleo em águas profundas: o caso da PETROBRAS**. 2001. 121p. Dissertação (Mestrado) – COPPE, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2001.

ONO, R.T. **Estudo de viabilidade do transporte marítimo de contêineres por cabotagem na costa brasileira**. 2001. 132p. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo. São Paulo, 2001.

PAPADAKIS, N.; PERAKIS, A.N. Fleet deployment optimization models – Part 2. **Maritime Policy & Management**, 14:2, p.145-155, 1987.

PAPADAKIS, N.; PERAKIS, A.N. A nonlinear approach to the multiorigin, multidestination fleet deployment problem. **Naval Research Logistics**, 36, p.515-528, 1989.

PARDALOS, P.M.; RESENDE, M.G.C. **Handbook of applied optimization**. Oxford: Oxford University Press, 2002. 1095p.

PERAKIS, A.N.; BREMER, W.M. An operational tanker scheduling optimization system: background, current practice and model formulation. **Maritime Policy & Management**, 19:3, p.177-187, 1992.

PERAKIS, A.N.; JARAMILLO, D.I. Fleet deployment optimization for liner shipping Part 1. Background, problem formulation and solution approaches. **Maritime Policy & Management**, 18:3, p.183-200, 1991.

PERAKIS, A.N.; PAPADAKIS, N. Fleet deployment optimization models – Part 1. **Maritime Policy & Management**, 14:2, p.127-144, 1987.

PESENTI, R. Hierarchical resource planning for shipping companies. **European Journal of Operational Research**, 86, p.91-102, 1995.

PILOTS. Paranaguá. Página institucional da empresa de serviços de praticagem Paranaguá Pilots Serviços de Praticagem S/C Ltda. Disponível em: <<http://www.paranaguapilots.com.br/manobras/anexos/anexo3.shtml>>. Acesso em: 24 de jan. 2007.

POCHET, Y.; WOLSEY, L.A. **Production planning by mixed integer programming**. New York: Springer, 2006. 499p. (Springer Series in Operations Research and Financial Engineering)

POTVIN, J.Y.; ROUSSEAU, J.M. An exchange heuristic for routing problems with time windows. **Journal of the Operational Research Society**, 46, p.1433-1446, 1995.

POWEL, B.J.; PERAKIS, A.N. Fleet deployment optimization for liner shipping: an integer programming model. **Maritime Policy & Management**, 24:2, p.183-192, 1997.

PRANDSTETTER, M.; RAIDL, G.R. A variable neighborhood search approach for solving the car sequencing problem. In: Mini Euro Conference on VNS, 18th, Tenerife, 2005. Anais. Brussels: EURO, 2005.

PRTISKER, A.A.; WATTERS, L.J.; WOLFE, P.M. Multiproject scheduling with limited resources: a zero-one programming approach. **Management Science**, 16:1, p.93-108, 1969.

PSARAFTIS, H. Foreword to the focused issue on maritime transportation. **Transportation Science**, 33, p.1-2, 1999.

RANA, K.; VICKSON, R.G. A model and solution algorithm for optimal routing of a time-chartered containership. **Transportation Science**, 22:2, p.83-95, 1988.

RAVETTI, M.G.; MATEUS, G.R.; ROCHA, P.L.; PARDALOS, P.M. A scheduling problem with unrelated parallel machines and sequence dependent setups. **International Journal of Operational Research**, 2:4, p.380-399, 2007.

RITCHIE, G. **Practical introduction to anchor handling and supply vessel operations**. Houston, Texas: Oilfield Publications Limited, 2004. 186p.

RONEN, D. Ship scheduling: the last decade. **European Journal of Operational Research**, 71, p.325-333, 1993.

RONEN, D. Short-term scheduling of vessels for shipping bulk or semi-bulk commodities in a single area. **Operations Research**, 34, p.164-173, 1986.

ROY, T.J.V.; WOLSEY, L.A. Valid inequalities and separation for uncapacitated fixed charge networks. **Operations Research Letters**, 4:3, p.105-112, 1985.

SAVELSBERGH, M.W.P. The vehicle routing problem with time windows: minimizing route duration. **ORSA Journal on Computing**, 4, p.146-154, 1992.

SHERALI, H.D.; AL-YAKKOB, S.M.; HASSAN, M. Fleet management models and algorithms for an oil-tanker routing and scheduling problem. **IIE Transactions**, 31, p.395-406, 1999.

SHINTANI, K.; IMAI, A.; NISHIMURA, E.; PAPADIMITRIOU, S. The container shipping network design problem with empty container repositioning. **Transportation Research Part E**, 43, p.39-59, 2007.

SOLOMON, M.M. On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. **Networks**, 16, p.161-174, 1986.

SOLOMON, M.M. Algorithms for the vehicle routing and scheduling problems with time window constraints. **Operations Research**, 35, p.254-265, 1987.

TAILLARD, É.; BADEAU, P.; GENDREAU, M.; GUERTIN, F.; POTVIN, J.Y. A tabu search heuristic for the vehicle routing problem with soft time windows. **Transportation Science**, 31:2, p.170-186, 1997.

THOMPSON, P.M.; ORLIN, J.B. Cyclic transfer algorithms for multivehicle routing and scheduling problems. **Operations Research**, 41, p.935-946, 1993.

WILLIAMS, H. P. **Model building in mathematical programming**. John Wiley & Sons, Inc., 1999. 354p.

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

XINLIAN, X.; TENGFEI, W.; DAISONG, C. A dynamic model and algorithm for fleet planning. **Maritime Policy & Management**, 27:1, p.53-63, 2000.