O problema de roteamento e programação de navios com coleta e entrega na indústria de petróleo: modelagem e métodos de solução exatos

Maria Gabriela Stevanato Furtado

O problema de roteamento e programação de navios com coleta e entrega na indústria de petróleo: modelagem e métodos de solução exatos [1]

Maria Gabriela Stevanato Furtado

Orientador: ***Prof. Dr. Reinaldo Morabito Neto***

Co-Orientador: ***Prof. Dr. Pedro Augusto Munari Junior***

Tese apresentada ao Departamento de Engenharia de Produção da Universidade Federal de São Carlos, como parte dos requisitos para obtenção do título de Doutor em Engenharia de Produção.

UFSCar - São Carlos

Maio/2016

[1]Este trabalho foi financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) processo número 2014/22542-2, pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e pela Agência Nacional do Petróleo (ANP).

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária UFSCar
Processamento Técnico
com os dados fornecidos pelo(a) autor(a)

Furtado, Maria Gabriela Stevanato
F992p O problema de roteamento e programação de navios com coleta e entrega na indústria de petróleo : modelagem e métodos de solução exatos / Maria Gabriela Stevanato Furtado. -- São Carlos : UFSCar, 2016.
189 p.

Tese (Doutorado) -- Universidade Federal de São Carlos, 2016.

1. Roteamento e programação de navios. 2. Problemas de coleta e entrega. 3. Indústria petrolífera. 4. Métodos exatos. I. Título.

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Engenharia de Produção

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de tese de doutorado do(a) candidato(a) Maria Gabriela Stevanato Furtado, realizada em 01/04/2016:

Prof. Dr. Reinaldo Morabito Neto
(UFSCar)

Prof. Dr. Pedro Augusto Munari Junior
(UFSCar)

Prof. Dr. André Bergsten Mendes
(USP)

Prof. Dr. Eduardo Uchoa Barboza
(UFF)

Prof. Dr. Geraldo Robson Mateus
(UFMG)

Profa. Dra. Vitória Maria Miranda Pureza
(UFSCar)

Certifico que a sessão de defesa foi realizada com a participação à distância do membro Prof. Dr. Eduardo Uchoa Barboza e, depois das arguições e deliberações realizadas, o participante à distância está de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa do(a) aluno(a) Maria Gabriela Stevanato Furtado.

Prof. Dr. Reinaldo Morabito Neto
Presidente da Comissão Examinadora
(UFSCar)

Aos meus amados pais Eduardo e Cristina,

e aos meus queridos avós Nizio e Neguinha.

"A persistência é o menor caminho do êxito" (Charles Chaplin).

Agradecimentos

É difícil expressar em palavras tudo o que sinto por estes quatro anos de doutorado. Não são muitos que conhecem ou já vivenciaram o quão difícil é obter o título de doutor, e como diz meu orientador "não existe doutorado fácil", e este com certeza não foi fácil. O principal conselho a ser vivenciado na academia é "não pirar", respirar fundo e seguir em frente. Sendo assim, o início dos meus agradecimentos se dará com um trecho de uma música do Raul Seixas: "Queira, basta ser sincero e desejar profundo, você será capaz de sacudir o mundo, vai...Tente outra vez". Muitas vezes quando eu saía desanimada da federal ou quando meu doutorado não se desenvolvia da maneira como planejado, era esta música que eu escutava e que me fazia constatar que eu precisava continuar tentando, e que um dia eu iria conseguir. Pois bem, de tanto tentar eu consegui! E para chegar até aqui eu tive a ajuda e o apoio de muitas pessoas, então estas pessoas que serão citadas com o maior carinho neste texto e eu espero que todas elas se sintam especiais com as palavras que deixo abaixo.

Primeiramente, eu gostaria de agradecer a Deus, ele que me deu forças quando eu mais precisei e que me ajudou a superar cada obstáculo e dificuldade. É a fé que me fez acreditar que tudo daria certo.

Aos meus pais Eduardo e Cristina, eles que sempre estiveram ao meu lado, apoiando as minhas decisões e me ajudando em absolutamente tudo que precisei, seja rezando por mim, ascendendo muitas velas, vindo almoçar aos domingos nos muitos finais de semana que eu fiquei em São Carlos trabalhando, seja por me ajudar financeiramente, me amar e acreditar que eu era capaz. Sei que é clichê falar que tenho os melhores pais, mas é verdade, eu tenho. O meu muito obrigada registrado aqui publicamente, eu amo vocês e sou muito grata por tudo que fizeram por mim.

Aos meus queridos orientadores, Reinaldo e Pedro. Eu iniciei meu doutorado sem saber se estava fazendo o certo, em uma época um pouco conturbada. Hoje eu vejo que ter voltado para a federal foi o melhor que poderia ter me acontecido, e isto se deve em grande parte a vocês, meus orientadores. Reinaldo, eu agradeço por você compartilhar comigo todas as suas ideias brilhantes, agradeço ainda a todas as suas palavras de con-

forto quando, para mim, tudo estava perdido, por confiar em mim e no meu trabalho e principalmente, por me orientar. Você sempre encontrou as melhores saídas para os problemas que surgiram ao longo deste doutorado. Você é um profissional brilhante, o qual eu tive o prazer de trabalhar. Pedrinho, eu serei eternamente grata por tudo o que você fez por mim. Só eu sei o quanto que você me ajudou neste doutorado. Você que tinha acabado de entrar na UFSCar como professor, me aceitou como sua aluna de prontidão, sem saber muito bem o que te esperava (acredite, eu também não sabia!). Só eu sei, as muitas vezes em que trabalhamos juntos na federal, ou às vezes em que eu fui até na sua casa para você me ajudar no código. Pedrinho, você não tinha obrigação alguma de fazer o que fez, mas mesmo assim estava lá, me ajudando e me orientando. Você vibrou comigo em cada pequena conquista deste doutorado, ficou feliz ou triste em muitos momentos e principalmente, sempre acreditou em mim. Espero continuar a parceira com vocês dois, parceria na vida e também no trabalho. Espero ainda que, um dia, eu possa ser uma orientadora tão boa quanto vocês foram para mim. Eu sinto muito orgulho de ser aluna de vocês. Muito obrigada por tudo!!!

Agradeço aos meus avós, Deonidio (Nizio) e Maria Aparecida (Neguinha). Eles que não tinham ideia do que eu fazia, mas mesmo assim acreditavam e sentiam muito orgulho de mim. Eles que me esperavam a cada sexta-feira que eu voltava para a Barra, e que se despediam de mim com um café na segunda-feira bem cedinho. Lembro bem quando consegui minha primeira bolsa do doutorado, eles ficaram tão felizes por mim que chego a me emocionar. Eu perdi os dois durante este doutorado, mas tenho a certeza que onde quer que estejam continuam vibrando com cada conquista minha.

Agradeço também à toda a minha família, tios, tias, primos, primas, cunhado e avó Maria Helena e em especial a minha irmã Fernanda, ela que é Engenheira de Produção e que nunca gostou de PO, mas que sempre esteve ao meu lado neste doutorado. Mary, obrigada pelas palavras, pela força, amizade e porque não, pelo dinheiro. Eu tenho muito orgulho de ser sua irmã, você me inspira e eu espero que um dia eu possa ser bem-sucedida como você é!

Aos envolvidos no projeto Petrobras, Denise, Roberto, Vitória, Bruno, Amélia, Paulo, Jonas e em especial ao Vinícius. Vini, você é um cara inspirador e um ótimo companheiro de trabalho. Obrigada pela palavra sempre amiga e por todo o conhecimento que foi transmitido a mim.

Aos companheiros de trabalho do Laboratório de Pesquisa Operacional: Alyne, Arthur, Karim, Pedro, Aldair, Juan, Tamara, Cesar e Admilson. Agradeço também aos amigos

que a vida me deu, Claudinha, Victor, Cherri, Marcão, Marcinho, Betina, Chuck (Rodrigo), Arthur, Leonardo, Caroline e Mayara. Cada um de vocês tem um espaço especial dentro do meu coração.

Agradeço as minhas mestras preferidas, Camila e Gisele. Cada uma de uma área diferente, humanas, biológicas e eu, de exatas, mas nos entendemos como ninguém. Vocês que fazem parte da minha vida há muito tempo, e que eu admiro e tanto amo. Obrigada por sempre atenderem aos meus pedidos de socorro nos momentos difíceis e por ficarem felizes com a minha felicidade.

Agradeço a todos os professores que de alguma forma contribuíram para a minha formação acadêmica e a todos os funcionários do DEP/UFSCar. Por fim, agradeço ao apoio financeiro da Petrobras (Agência Nacional de Petróleo (ANP)), Capes (Coordenação de aperfeiçoamento de pessoal de nível superior) e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) - processo número 2014/22542-2 .

Resumo

O objeto de estudo deste trabalho é o problema de roteamento e programação de navios com coleta e entrega e janelas de tempo na indústria petrolífera. Foi realizado um estudo de caso com uma empresa petrolífera brasileira que produz óleo cru em plataformas *offshore*, isto é, localizadas no oceano e os transporta até os terminais localizados na costa brasileira. Então, foi proposto um modelo de programação inteira mista para representar o problema adequadamente e para isso, foi necessária uma análise detalhada do problema real, com o intuito de conhecer todas as suas características e considerar hipóteses simplificadoras. Desta maneira, ao problema de coleta e entrega e janelas de tempo da literatura foram agregadas outras restrições específicas do problema do estudo de caso como, por exemplo, múltiplos depósitos, restrições de atracação dos navios, calado flexível e posicionamento dinâmico. Além disso, a frota de navios é heterogênea em relação à capacidade, LOA (*length overall*), posicionamento dinâmico e velocidade. Na prática, em geral não existem navios iguais. Este problema pode ser representado como um modelo de otimização combinatória que pertence à classe NP-difícil e sua solução é bastante desafiadora na prática em função do tamanho dos problemas reais. Depois, foram propostos vários métodos do tipo *branch-and-cut* baseados em modelos com variáveis de 2 e 3-índices para problemas de roteamento com coleta e entrega e janelas de tempo para resolver especificamente o problema da empresa brasileira. E por fim, foi proposto um método do tipo *branch-and-price*, o qual abrange todas as características do problema da indústria petrolífera. Em síntese, as principais contribuições desta tese referem-se ao estudo e modelagem deste problema na prática, e a proposta e desenvolvimento de métodos de solução exatos para resolvê-lo, baseados em *branch-and-cut* e *branch-and-price*. O desempenho do modelo matemático em *softwares* de otimização e também dos métodos exatos propostos foi verificado usando-se exemplares reais fornecidos pela empresa. Os resultados mostram que essas abordagens podem ser efetivas para resolver problemas de tamanho moderado em situações reais.

PALAVRAS CHAVE: **Roteamento e programação de navios, Problemas de coleta e entrega, Indústria petrolífera, Métodos exatos, Método *branch-and-cut*, Método *branch-and-price***.

Abstract

The object of this study is the routing and scheduling problem of vessels with pickup and delivery and time windows in the oil industry. A case study was performed in a Brazilian oil industry that produces crude oil in offshore platforms, that is, located in the ocean, and transports to the terminals located in the Brazilian coast. Then, it was proposed a mixed integer model to represent the problem adequately and for this, a detailed analysis of the real problem in order to know all its characteristics and consider some simplifying assumptions. Therefore, to the pickup and delivery problem with time windows present in the literature were aggregated other specific restrictions of the case study, for example, multiple depots, ship mooring restrictions, flexible draft and dynamic positioning. Besides that, the fleet is heterogeneous related to capacity, LOA (length overall), dynamic positioning and velocity. In practice, in general there are no identical vessels. This problem can be represented as a combinatorial optimization model, which belongs to the NP-hard class and its solution is a challenging in practice depending on the size of the real problems. Then, were proposed several exact branch-and-cut methods based on models with 2 and 3-index variables for routing problems with pickup and delivery and time windows to solve specifically the Brazilian oil industry problem. Finally, we proposed a branch-and-price method, which includes all characteristics of the problem in oil industry. In summary, the main contributions of this thesis are related to the study and modeling of this problem in practice, and the proposal and development of exact solution methods to solve it, based on branch-and-cut and branch-and-price. The performance of the mathematical model in optimization software and the exact methods were verified using a real data set provided by the company. Results show that these approaches may be effective to solve problems of moderate size in real situations.

KEYWORDS: **Routing and scheduling of vessels, Pickup and delivery problem, Oil industry, Exact methods, Branch-and-cut method, Branch-and-price method**.

Sumário

Lista de Tabelas

Lista de Figuras

1 Introdução

A economia mundial progrediu de maneira significativa com a disseminação da globalização. Com o aumento dos serviços de transporte de produtos pelo mundo, a logística passou a ser relevante para que os custos associados ao transporte nas empresas não se tornassem excessivos. Sendo assim, a eficiência do transporte se tornou importante para a sociedade (CHRISTIANSEN et al., 2007). Neste processo de distribuição de mercadorias, o transporte é fator fundamental e quando não existe uma rede de transportes eficiente, o comércio se limita aos centros de produção (BALLOU, 2010). O setor de transporte marítimo não é diferente; é necessária uma logística eficaz para que o transporte de produtos seja otimizado, evitando gastos desnecessários e aproveitando da melhor maneira possível os recursos disponíveis.

O transporte marítimo cresceu cerca de 25% desde 1980 e a indústria marítima passou a ter o monopólio no transporte de grandes volumes entre continentes (CHRISTIANSEN et al., 2007). Em Christiansen et al. (2004), os autores citam que esta atividade provavelmente irá aumentar no futuro, resultando em grupos internacionais colaborando entre si e se unindo para a melhor extração dos recursos naturais disponíveis. Por ser um setor muito importante para a economia, é fundamental que estudos sobre o transporte marítimo sejam realizados a fim de aprimorar sua eficiência e diminuir os custos associados. Nos últimos anos, os investimentos no setor de transportes têm sido maiores (CHRISTIANSEN et al., 2004) e, do ponto de vista acadêmico, os problemas relacionados a este setor têm recebido maior atenção (HOFF et al., 2010).

Os problemas associados ao transporte marítimo podem ser divididos em três classes: problemas estratégicos, táticos e operacionais (VATN, 2007). As decisões estratégicas têm como objetivo, por exemplo, selecionar rotas comerciais e de mercado, projetar navios, programar sistemas de transporte e redes e especificar o tamanho da frota e dos terminais. As decisões táticas dizem respeito, por exemplo, à alocação de berços de atracação e guindastes, planejar e administrar os contêineres e, em alguns casos, ao roteamento e programação de navios (para longas distâncias). As decisões operacionais têm como ob-

jetivo, por exemplo, selecionar a velocidade do navio, decidir sobre os tempos de espera e também ao roteamento e programação de navios (para curtas distâncias).

Em relação ao setor petrolífero, este tem crescido em todo o mundo e em especial no Brasil. A capacidade de produção brasileira de petróleo é de cerca de 2,1 milhões de barris diários. Segundo o Ministério do Desenvolvimento, Indústria e Comércio Exterior (MDIC, 2011), as exportações de petróleo bruto em dezembro de 2011 somaram 3,54 milhões de toneladas. As maiores reservas de petróleo hoje estão na plataforma continental em águas profundas e ultraprofundas. Segundo Petrobras (2012), o Brasil tem cerca de 65% da área de seus blocos exploratórios *offshore*, ou seja, plataformas localizadas no oceano, em profundidades de água de aproximadamente 400 metros. Em consequência, nos últimos anos, a empresa tem aumentado suas atividades de perfuração exploratória em águas cada vez mais profundas.

O objeto de estudo deste trabalho é o roteamento e a programação dos navios que transportam o óleo cru das plataformas *offshore* até os terminais da costa brasileira. Este problema pode ser considerado como um problema de decisões operacionais devido as distâncias envolvidas serem relativamente curtas. O problema trata de um conjunto de solicitações (demandas), cujos atributos definem total ou parcialmente o translado de um produto a partir de uma origem para um destino, com prazos definidos em janelas de tempo, que impõem intervalos rígidos para permissão de retiradas do produto e entrega do mesmo.

1.1 Objetivos

Os principais objetivos desta tese são:

- Desenvolver um estudo de caso em uma empresa brasileira e conhecer a fundo esse problema real, e propor modelos de otimização combinatória para representar adequadamente o roteamento e a programação de navios com coleta e entrega e janelas de tempo, de modo a contemplar os requisitos da empresa brasileira que produz e transporta óleo cru das plataformas até os terminais. Inicialmente os modelos serão implementados em uma linguagem de modelagem e resolvidos com o uso de *softwares* de otimização de propósito geral como, por exemplo, o IBM ILOG *CPLEX* (ILOG, 2013);

- Pesquisar e desenvolver métodos de solução específicos do tipo *branch-and-cut* e

branch-and-price (DESROSIERS; LÜBBECKE, 2010) para resolver os modelos desenvolvidos, os quais permitem explorar características especiais do problema da empresa brasileira. Estes métodos têm sido amplamente estudados na literatura de pesquisa operacional e têm se mostrado bastante promissores quando aplicados a problemas de roteamento de veículos (CORDEAU, 2006; ROPKE et al., 2007; ROPKE; CORDEAU, 2009; QU; BARD, 2015).

- Analisar o desempenho e aplicabilidade dos métodos desenvolvidos usando exemplares reais disponibilizados pela empresa brasileira, assim como exemplares gerados aleatoriamente. Pretende-se mostrar o potencial de aplicação dessas abordagens para resolver o problema estudado nesta tese.

1.2 Justificativa

Os problemas de roteamento e programação de navios têm relevância técnica e econômica e além disso, podem envolver diversas características específicas e diferentes abordagens para sua solução. Diante disto, diversos métodos de solução podem ser aplicados neste contexto. Conforme revisão bibliográfica, não foram encontrados trabalhos na literatura que representassem completamente o problema de coleta e entrega de óleo cru na indústria petrolífera brasileira. Há trabalhos que abordam o tema, mas apenas considerando algumas características isoladamente como, por exemplo, coleta e entrega com frota heterogênea ou coleta e entrega com janelas de tempo e múltiplos depósitos. Ou seja, não foram encontrados trabalhos que contemplassem de forma integrada todas as características envolvidas no problema do estudo de caso: janelas de tempo, múltiplos depósitos, frota heterogênea, restrições de atracação dos navios, calado flexível, posicionamento dinâmico, dentre outras características. Portanto, existe uma lacuna na literatura na linha de pesquisa explorada neste trabalho. Os trabalhos presentes na literatura abordam o problema de coleta e entrega com janelas de tempo, porém ou consideram múltiplos depósitos ou consideram frota heterogênea. Este trabalho considera todas estas características juntamente com outras adicionais relacionadas ao problema do estudo de caso.

Uma contribuição desta tese é o estudo e desenvolvimento de modelos que consideram restrições específicas encontradas no transporte de petróleo no Brasil, baseados no modelo clássico de roteamento e programação de veículos com coleta e entrega e janelas de tempo (*pickup and delivery problem with time windows - PDPTW*). Outra contribuição desta

tese se dá pela pesquisa e proposição de métodos exatos desenvolvidos especificamente para resolver o problema do estudo de caso e baseados nos métodos *branch-and-cut* e *branch-and-price*. Foram necessárias várias modificações nos métodos clássicos da literatura para que fosse possível a resolução do problema de coleta e entrega na indústria petrolífera como, por exemplo, adaptação de algumas desigualdades válidas, modificação e incluisão de restrições nos modelos matemáticos e consideração de todas as características que envolvem o problema do estudo de caso dentro de cada método proposto. Também desenvolvemos um estudo de caso com a importante colaboração da empresa brasileira para evidenciar o potencial de aplicação das abordagens desenvolvidas.

1.3 Metodologia

O método de pesquisa aqui utilizado é a modelagem/simulação, especificamente pesquisa empírica normativa (BERTRAND; FRANSOO, 2002) fortemente baseado em situações reais. Este tipo de pesquisa visa o desenvolvimento de políticas, estratégias e ações que melhorem a situação corrente. Esta pesquisa baseia-se em modelos que prescrevem uma decisão para o problema, baseados em programação matemática (MORABITO; PUREZA, 2010). Esta tese se dá pela investigação e análise de vários tipos de modelos para o problema de coleta e entrega da empresa do estudo de caso. O primeiro modelo, baseado em Cordeau (2006), considera frota heterogênea, janelas de tempo, restrições de capacidade, dentre outras características. Este modelo possui um número polinomial de restrições e variáveis, e exemplares de tamanho moderado podem ser resolvidos por *softwares* de otimização de propósito geral, como o *CPLEX*. Porém, pode ser bastante ineficiente em exemplares de maior porte, devido a certas características do modelo, como simetria e relaxação linear fraca. Outros dois modelos investigados são baseados na modelagem de Ropke et al. (2007), conforme apresentado no próximo capítulo. Estas modelagens possuem um número de restrições que cresce exponencialmente em função do número de coletas e entregas do problema. A implicação deste fato é que é praticamente intratável computacionalmente explicitar todas as suas restrições. Sendo assim, deve-se considerar apenas um subconjunto limitado destas restrições e adicioná-las ao modelo apenas quando for necessário, utilizando algoritmos de separação apropriados, como é feito no método *branch-and-cut*.

O método *branch-and-cut* é uma combinação do método *branch-and-bound* com o uso de planos de corte. Em cada nó da árvore de busca acrescentam-se desigualdades válidas que são violadas pela solução do problema relaxado da iteração correspondente

(utilizando algoritmos de separação). Além disso, tem-se as operações padrão do *branch-and-bound* para ramificação e também eliminação de nós por meio de limitantes. Em Ropke et al. (2007), os autores testam uma série de desigualdades válidas e assim, propõem dois métodos do tipo *branch-and-cut* para sua resolução. Porém, estes cortes são testados para frota homogênea e um único depósito inicial e final, para todos os veículos. Essas desigualdades válidas são então estendidas para o problema da empresa brasileira e incorporadas aos métodos *branch-and-cut* propostos. Outro modelo com variáveis com 2-índices é testado neste contexto do método *branch-and-cut*, no qual o número de restrições também cresce exponencialmente em relação ao número de pares de coleta e entrega. Este modelo é baseado em Furtado et al. (2015) e mostrado no Capítulo 5. Além disso, o método *branch-and-cut* é testado com outros modelos, com variáveis com 3-índices, o que também está explicitado no Capítulo 5.

A desvantagem de se usar um método *branch-and-cut* neste contexto é que a relaxação linear do modelo ainda pode ser fraca, mesmo após a inserção de cortes. Desta maneira, o método *branch-and-cut* pode se tornar ineficiente. Diante desta dificuldade, técnicas de decomposição têm sido aplicadas na literatura relacionada a este contexto, as quais resultam em formulações com relaxações lineares que resultam em melhores limitantes. Uma técnica muito utilizada e bem sucedida é a decomposição de *Dantzig-Wolfe* (DANTZIG; WOLFE, 1960; VANDERBECK, 2000). Esta estratégia decompõe uma formulação em duas partes: uma chamada de problema mestre (com número exponencial de variáveis em relação ao número de pares de coleta e entrega) e outra de subproblema (que é responsável pela geração das colunas do problema mestre). No caso do roteamento e programação de veículos, o problema mestre resultante é conhecido como a formulação de particionamento de conjuntos (*set partitioning*), que é o último modelo explorado neste trabalho. Para evitar a enumeração de todas as colunas do problema mestre, utiliza-se o método de geração de colunas. Ao se combinar o método *branch-and-bound* com o método de geração de colunas, obtêm-se o método *branch-and-price*, conforme detalhado na Seção 3.3 (PESSOA et al., 2008; ROPKE; CORDEAU, 2009; DESROSIERS; LÜBBECKE, 2010). A partir do *branch-and-price* descrito em Ropke e Cordeau (2009) foi desenvolvido um método específico para o nosso problema.

Este texto está estruturado da seguinte maneira: no Capítulo 2 é apresentada uma breve revisão bibliográfica para os problemas de roteamento e programação de veículos. No Capítulo 3 é apresentada uma revisão específica da literatura para os métodos exatos desenvolvidos nesta tese. O problema estudado de roteamento e programação de navios na indústria petrolífera, o modelo matemático desenvolvido para o estudo de caso e os

resultados computacionais para o modelo são apresentados no Capítulo 4. No Capítulo 5 é apresentado o método de solução *branch-and-cut*, com as adaptações feitas para o estudo de caso e os resultados computacionais obtidos. No Capítulo 6 o método *branch-and-price* desenvolvido especificamente para a indústria petrolífera é detalhado, mostrando todas as suas características e particularidades de implementação. Além disso, os resultados computacionais para este método também são mostrados no Capítulo 6. Por fim, no Capítulo 7 são apresentadas as conclusões desta tese e também são discutidas algumas perspectivas para pesquisas futuras.

2 *Problemas de roteamento e programação de veículos*

O problema clássico de roteamento de veículos (*vehicle routing problem* - VRP) consiste em designar rotas para que seja entregue toda a demanda de um conjunto de nós. Mais especificamente, cada veículo sai do depósito comum a todos os veículos, visita todos os nós da sua rota, e retorna ao mesmo depósito. Os objetivos podem envolver minimizar os custos relacionados ao transporte, minimizar o número de veículos, balancear as rotas, dentre outros. O termo programação de veículos pode ser usado nesta classe de problemas quando há imposições quanto aos instantes de tempo em que cada veículo deve chegar e sair de cada nó. Diversas variantes do VRP foram propostas na literatura e impõem particularidades, como restrições de capacidade (*Capacitated vehicle routing problem* - CVRP), janelas de tempo (*Vehicle routing problem with time windows* - VRPTW), restrições de coletas e entregas (*Pickup and delivery problem* - PDP), múltiplos depósitos (*Multi-depot vehicle routing problem* - MDVRP), entre outras variações (BERBEGLIA et al., 2007; DESROCHERS et al., 1990; GOLDEN et al., 2008; LAPORTE, 2009; PARRAGH et al., 2008b).

A rede que contempla as rotas é geralmente descrita por um grafo, cujos nós correspondem aos pontos a serem visitados e ao depósito, enquanto os arcos do grafo representam a possibilidade de se transitar entre dois nós. Este grafo pode ser direcionado ou não-direcionado, dependendo se o veículo pode percorrer um arco em uma única direção ou ambas, respectivamente. A cada arco está associado um custo, que geralmente é proporcional à distância mínima e/ou tempo de viagem entre os nós, que pode ou não depender do tipo de veículo. Além disso, podem ter janelas de tempo (períodos de tempo em que cada nó deve ser visitado) e tempos de carregamento e descarregamento da carga que podem depender dos veículos. Quando não é possível satisfazer a demanda de todos os nós, algumas variantes do problema permitem que a carga seja fracionada (ARCHETTI et al., 2008) ou alguns nós não sejam atendidos, desde que estas penalidades sejam contabilizadas na função objetivo.

As rotas devem satisfazer algumas restrições operacionais como, por exemplo, que a capacidade do veículo não pode ser excedida e o cliente deve ser visitado dentro da sua janela de tempo. Restrições de precedência impõem a ordem da visita dos nós (LAPORTE, 2009). Quanto ao depósito, este pode ser comum a todos os veículos ou pode existir mais de um depósito (BETTINELLI et al., 2011). A capacidade e outras características do veículo podem ser diferentes (frota heterogênea) e cada veículo pode ter o seu interior subdividido em compartimentos menores. Podem haver custos associados à utilização dos veículos e, além disso, um subconjunto de veículos pode, por exemplo, ter restrições em atender certos nós.

Algumas revisões da literatura a respeito dos problemas de roteamento e programação de veículos podem ser encontradas em Desrochers et al. (1990), Laporte e Nobert (1987), Berbeglia et al. (2007) e Laporte (2009). Alguns dos principais livros na área são Barnhart e Laporte (2007), Pereira e Tavares (2008), Golden et al. (2008) e Toth e Vigo (2014). Uma revisão recente sobre o problema de roteamento de veículos com frota heterogênea pode ser encontrada em Koç et al. (2015). Neste capítulo, revisam-se algumas modelagens matemáticas para o VRP, com modelos com variáveis de 2 e 3-índices e um modelo de particionamento de conjuntos, que serviram de pontos de partida para a modelagem do problema desta tese.

2.1 Problemas clássicos

O problema de roteamento de veículos com restrições de capacidade, ou seja, o VRP capacitado (CVRP - *capacitated vehicle routing problem*) (LAPORTE; NOBERT, 1987) consiste em designar rotas de custo mínimo a veículos com capacidade limitada. No problema clássico, as demandas dos nós são determinísticas, ou seja, conhecidas a *priori*, a frota é homogênea e há somente um único depósito central. A notação apresentada nesta seção é a mesma utilizada por Cordeau (2006), Ropke et al. (2007) e Ropke e Cordeau (2009).

O CVRP consiste em encontrar as rotas dos veículos com mínimo custo total de tal forma que cada veículo comece e termine no depósito inicial e final, respectivamente, cada nó seja visitado uma única vez e o somatório das demandas dos nós visitados numa mesma rota não exceda a capacidade do veículo. O CVRP pertence à classe NP-difícil e é uma generalização do problema do caixeiro viajante (*Traveling Salesman Problem* - TSP), que envolve em um único veículo para visitar todos os nós e não há restrições de capacidade (GAVISH; GRAVES, 1979).

O VRPTW (*vehicle routing problem with time windows*) é uma variação do CVRP em que, além das restrições relacionadas à capacidade, há também restrições que limitam o tempo em que o veículo pode chegar e sair de um determinado nó. Sendo assim, a cada nó i está associada uma janela de tempo representada pelo intervalo $[e_i,l_i]$, em que e_i é o instante de tempo inicial em que o veículo pode iniciar o serviço no nó i e l_i é o instante máximo em que o veículo pode iniciar o serviço no nó i. O VRPTW consiste em determinar as rotas dos veículos disponíveis com menor custo, de modo que cada veículo comece e termine no depósito, cada nó seja visitado uma única vez, o somatório das demandas dos nós numa mesma rota não exceda a capacidade do veículo e o serviço de cada nó se inicie dentro da janela de tempo (DESROCHERS et al., 1988). Vários autores trataram este problema e o resolveram por métodos exatos e também por métodos heurísticos. Em Bräysy e Gendreau (2005) e Bräysy et al. (2005b), os autores fazem uma revisão de heurísticas para resolver o VRPTW baseadas na construção da rota, busca local e algoritmos evolutivos. São descritas as características de cada método, bem como os resultados computacionais. Em Baldacci et al. (2010), os autores apresentam um *framework* na resolução de variações do VRP através de métodos exatos.

Outra característica dos problemas de roteamento é a permissão para fracionar a carga coletada ou entregue no nó. Isto é discutido em Archetti et al. (2008), em que os autores analisam as vantagens e desvantagens de se fracionar ou não as cargas. Ainda na linha de cargas fracionárias, podemos citar Belfiore e Yoshizaki (2009), Moreno et al. (2010) e Archetti e Speranza (2012). Revisões da literatura sobre o VRP e suas variações são encontradas em Eksioglu et al. (2009), Nagy e Salhi (2007), Pillac et al. (2013) e Potvin (2009).

Há outras variações do VRP clássico como, por exemplo, o agrupamento e alocação das entregas em que a distância entre os nós é desconsiderada, pois o custo não tem relevância na função objetivo (REIS; CUNHA, 2011). A frota heterogênea é considerada em Penna et al. (2013), em que uma heurística de busca local é aplicada ao problema. Outra variação do VRP é considerar restrições de acessibilidade para os nós com uma frota heterogênea e janelas de tempo; cada veículo pode percorrer várias rotas por dia, e cada veículo deve começar e terminar sua rota num mesmo depósito. Para tanto, um algoritmo de geração de colunas é proposto (SEIXAS, 2013). Outros trabalhos que envolvem a resolução do VRP aparecem em Pureza e França (2001), Branchini et al. (2009), Pureza et al. (2012), Seixas (2013).

No problema de roteamento de veículos com *backhauls*, o conjunto de nós N é subdi-

vidido em dois outros conjuntos: nós em que ocorre coleta e nós em que ocorre a entrega de produtos. Os produtos entregues estão inicialmente no depósito inicial e os produtos coletados são os presentes em cada nó de coleta. Por exemplo, um caminhão que entrega refrigerantes pode também coletar as embalagens que são retornáveis e levá-las novamente para a fábrica. Nesta variação, restrições de precedência são necessárias para garantir que todos os nós de entrega sejam visitados antes dos nós de coleta. Rotas somente com nós de coleta não são permitidas e, como nas demais extensões, cada veículo sai do depósito e retorna ao mesmo. Deve-se garantir que a capacidade do veículo seja respeitada e que cada nó seja visitado uma única vez (KÜÇÜKOGLU; ÖZTÜRK, 2013).

No problema de roteamento de veículos com coleta e entrega (PDP - *pickup and delivery problem*), a cada nó i temos que uma quantidade q_i deve ser coletada; esta quantidade deve ser entregue no nó $n+i$, que possui demanda de $q_{n+i} = -q_i$, que representa a respectiva entrega. Note que a diferença entre os problemas do tipo *backhauls* e o PDP é que no primeiro o veículo não coleta em um respectivo nó para entregar esta quantidade em outro nó já pré-definido. Em um problema do tipo *backhaul*, os produtos que são entregues estão inicialmente no depósito, e os produtos que são coletados vêm dos próprios clientes. Já no PDP, a cada coleta do nó i está associada uma demanda que deve ser entregue no nó $n + i$. Neste tipo de problema, além das restrições impostas nas outras extensões, temos que a coleta deve ser realizada antes da entrega e que ambas devem estar na rota de um mesmo veículo. Alguns trabalhos desta variação do VRP são Berbeglia et al. (2007), Ropke et al. (2007), Nowak et al. (2008), Parragh et al. (2008), Pureza e Laporte (2008), Ropke e Cordeau (2009), Nowak et al. (2009) e Hennig et al. (2012).

Há 3 formulações matemáticas básicas do VRP: formulação de fluxo de veículos (*vehicle flow formulation*), formulação de fluxo de mercadorias (*commodity flow formulation*) e problema de particionamento de conjuntos. Nos modelos de fluxo de veículos (DESROCHERS; LAPORTE, 1991), as variáveis são inteiras e contabilizam o número de vezes que o arco é percorrido pelo veículo. Este tipo de modelo é mais comum em casos em que o valor da função objetivo é dado como o somatório dos custos associados aos arcos. Além disso, a relaxação linear desta formulação é dita fraca, ou seja, o valor ótimo obtido resolvendo-se a relaxação linear do modelo está distante do valor da solução ótima inteira.

Nos modelos do tipo fluxo de mercadorias (GAVISH; GRAVES, 1979, 1982) as variáveis são inteiras e representam o fluxo que passa pelos caminhos feitos pelos veículos, ou seja, o fluxo de mercadorias em cada arco da rota do veículo. A relaxação linear tende a ser melhor que a dos modelos de fluxo de veículos. No problema de particionamento de

conjuntos (BALINSKI; QUANDT, 1964) existe um número exponencial de variáveis binárias, cada uma associada a um caminho factível diferente para os veículos. A relaxação linear deste tipo de problema é a mais apertada, porém o número de variáveis binárias cresce exponencialmente em relação ao número de nós, sendo inviável enumerar todas elas.

2.2 Problemas de Coleta e Entrega

Como já definido, no problema de coleta e entrega, um conjunto de veículos com capacidade limitada tem que satisfazer as solicitações dos clientes. Porém, neste caso, a cada solicitação está associado um nó de origem e um nó de destino (ou seja, a demanda está pareada) e o mesmo veículo deve coletar o pedido na origem pré-fixada e o entregar no destino pré-fixado. Além disso, ao contrário do que ocorre no VRP clássico, as demandas são positivas ou negativas, dependendo se o nó é de coleta ou entrega, respectivamente.

Neste contexto de coleta e entrega há muitas aplicações como, por exemplo, o transporte de pessoas, transporte de mercadorias de um centro de distribuição para supermercados da região, dentre outras. Semelhante ao VRPTW, podem haver janelas de tempo, resultando no PDPTW (*pickup and delivery problem with time windows*). No caso do transporte de pessoas, restrições adicionais são necessárias para melhorar a satisfação do cliente como, por exemplo, restrições de tempo limite de percurso usadas no *dial-a-ride problem* (DARP) (CORDEAU; LAPORTE, 2003; CORDEAU, 2006).

Tanto o PDP como o PDPTW são variações do VRP clássico também NP-difíceis. Desta maneira, a maioria dos trabalhos da literatura foca em heurísticas (GENDREAU et al., 1999; DESAULNIERS et al., 2002; ROPKE; PISINGER, 2006; BIANCHESSI; RIGHINI, 2007; NOWAK et al., 2008). Porém, também há uma vasta literatura dos problemas sendo resolvidos por métodos exatos, conforme resumidamente descrito na sequência.

2.2.1 Revisão da Literatura

O caso de coleta e entrega com um único veículo foi estudado primeiramente por Ruland e Rodin (1997). Neste artigo, os autores propõem um algoritmo *branch-and-cut* e quatro classes de desigualdades válidas. Estas desigualdades também podem ser aplicadas ao problema com janelas de tempo. Dentro desta mesma linha, Dumitrescu et al. (2010) estudam o TSP com coleta e entrega e modelam como um problema linear inteiro, cuja estrutura poliedral é analisada. Com isto, várias desigualdades válidas são introduzidas e testadas e um algoritmo *branch-and-cut* é apresentado. Outra análise importante é feita

em Berbeglia et al. (2012), em que os autores estudam a incorporação de rotas fixas ao problema de coleta e entrega e propõem um algoritmo *branch-and-cut* para sua resolução.

Para o problema de coleta e entrega com janelas de tempo e múltiplos veículos (PDPTW), Dumas et al. (1991) propõem um algoritmo exato baseado em geração de colunas e restrições de caminho mínimo para o subproblema. Este algoritmo pode ser utilizado também para múltiplos depósitos e frota heterogênea. Baldacci et al. (2011) estudam o PDPTW com restrições de precedência e pareamento. Os autores propõem um algoritmo baseado no modelo de particionamento de conjuntos e os resultados foram efetivos para exemplares razoavelmente grandes da literatura.

Para variações do PDP, também são encontrados trabalhos na literatura que exploram métodos exatos. Para o problema de coleta e entrega com múltiplos veículos, Lu e Dessouky (2004) propõem um algoritmo *branch-and-cut* com quatro classes de desigualdades válidas, em que as restrições de janela de tempo são ditas *softs*, isto é, podem ser violadas mediante um custo na função objetivo. Em outro trabalho, o problema de roteamento com coleta e entrega e localização é modelado por Karaoglan et al. (2011). Este problema consiste em encontrar localizações dos depósitos e designar as rotas dos veículos de modo que a demanda de cada coleta e entrega seja satisfeita pelo mesmo veículo. Os autores propõem um algoritmo *branch-and-cut* para sua resolução e implementam uma série de desigualdades válidas que são adaptadas para esta variação do problema de roteamento. Alguns *surveys* relevantes podem ser encontrados em Savelsbergh e Sol (1995), Desaulniers et al. (2002), Berbeglia et al. (2007), Cordeau et al. (2008), Parragh et al. (2008) e Parragh et al. (2008b).

O DARP consiste em uma variação do problema de coleta e entrega em que é necessário designar rotas aos n usuários. Este problema possui restrições adicionais como, por exemplo, tempo máximo de rota para cada usuário, dentre outras. Para o DARP, Cordeau (2006) apresenta uma formulação inteira-mista com restrições de capacidade, duração total da rota, janelas de tempo, pareamento das demandas e precedência. Um algoritmo *branch-and-cut* é desenvolvido e novas desigualdades válidas são propostas, as quais até então provadas válidas apenas para o TSP.

Ropke et al. (2007) e Ropke e Cordeau (2009) são trabalhos importantes e que servem de base para o problema desta tese. No primeiro, são propostos dois modelos para o problema de coleta e entrega com janelas de tempo baseados em formulações de 2-índices. Para tanto, a frota de veículos é ilimitada e homogênea e deve-se respeitar as restrições de capacidade dos veículos e janelas de tempo dos clientes. O objetivo é minimizar os custos

relacionados às viagens. Os autores apresentam os resultados de experimentos computacionais comparando as formulações com o modelo de 3-índices proposto por Cordeau (2006) para o mesmo problema. Os resultados mostram que o método proposto é capaz de resolver um número maior de exemplares até a otimalidade, utilizando menor tempo computacional. Em continuidade a este trabalho, Ropke e Cordeau (2009) propõem um método *branch-and-cut-and-price* para o problema de coleta e entrega de veículos. O método utiliza as desigualdades válidas apresentadas em Cordeau (2006), Ruland e Rodin (1997) e Ropke et al. (2007), e os resultados mostram que para exemplares grandes o método é capaz de resolver os problemas em tempo computacional razoável.

2.2.2 Formulações Matemáticas

Para a modelagem do problema de coleta e entrega, a mesma notação já apresentada para o problema de roteamento e programação de veículos é utilizada. Para tanto, vamos acrescentar algumas características e também variáveis para este problema. O modelo que é apresentado a seguir é baseado em Ropke e Cordeau (2009).

Seja n o número de pedidos que precisam ser atendidos. A rede $G(N,A)$ consiste em N, um conjunto de nós, e A, um conjunto de arcos, e temos que $N = \{0,...,2n+1\}$, em que 0 e $2n+1$ representam os depósitos inicial e final, respectivamente (que podem ter a mesma localização). Os conjuntos $P = \{1,...,n\}$ e $D = \{n+1,...,2n\}$ representam o conjunto dos nós de coletas e entregas, respectivamente. Para cada coleta associada ao nó i tem-se uma entrega associada ao nó $n+i$. A cada nó $i \in N$ e $n+i \in N$ temos uma demanda $q_i > 0$ e $q_{n+i} < 0$ e um tempo de serviço $d_i \geq 0$, satisfazendo $d_0 = d_{2n+1} = 0$, $q_0 = q_{2n+1} = 0$ e $q_{n+i} = -q_i$ para $i = 1,...,n$. K é o conjunto dos veículos e cada veículo $k \in K$ possui uma capacidade Cap_k. Uma janela de tempo $[e_i,l_i]$ está associada a cada nó $i \in P \cup D$, em que e_i e l_i representam os limites inferior e superior da janela de tempo, respectivamente. Os depósitos inicial e final também possuem suas janelas de tempo, que indicam os instantes que cada veículo pode sair e retornar ao depósito. O custo de viagem do nó i para o nó j no veículo k é denotado por c_{ijk} e o tempo de viagem entre os dois nós é dado por t_{ij}.

Para cada arco $(i,j) \in A$ e cada veículo $k \in K$, a variável de decisão x_{ijk} deve assumir valor 1 se o veículo k viaja diretamente do nó i para o nó j, e 0 caso contrário. Para cada nó $i \in N$ e cada veículo $k \in K$, definimos a variável de decisão B_{ik} como o tempo de início de serviço do veículo k no nó i. Seja Q_{ik} a variável que indica a carga do veículo k logo após visitar o nó i. O modelo com variáveis de 3-índices para o PDPTW pode ser

modelado da seguinte maneira:

$$(PDPTW1) \qquad \text{Min} \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} c_{ijk}x_{ijk} \tag{2.1}$$

s.a

$$\sum_{k\in K}\sum_{j\in N} x_{ijk} = 1 \qquad \forall i \in P \tag{2.2}$$

$$\sum_{j\in N} x_{ijk} - \sum_{j\in N} x_{n+i,j,k} = 0 \qquad \forall k \in K; i \in P \tag{2.3}$$

$$\sum_{j\in P} x_{0jk} = 1 \qquad \forall k \in K \tag{2.4}$$

$$\sum_{j\in N} x_{jik} - \sum_{j\in N} x_{ijk} = 0 \qquad \forall i \in P \cup D; k \in K \tag{2.5}$$

$$\sum_{i\in D} x_{i,2n+1,k} = 1 \qquad \forall k \in K \tag{2.6}$$

$$B_{jk} \geq (B_{ik} + d_i + t_{ij})\, x_{ijk} \qquad \forall i \in N; j \in N; k \in K \tag{2.7}$$

$$Q_{jk} \geq (Q_{ik} + q_j)\, x_{ijk} \qquad \forall i \in N; j \in N; k \in K \tag{2.8}$$

$$B_{ik} + d_i + t_{i,n+i} \leq B_{n+i,k} \qquad \forall i \in P; k \in K \tag{2.9}$$

$$e_i \leq B_{ik} \leq l_i \qquad \forall i \in N; k \in K \tag{2.10}$$

$$\max\{0,q_i\} \leq Q_{ik} \leq \min\{Cap_k, Cap_k + q_i\} \qquad \forall i \in N; k \in K \tag{2.11}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in N; j \in N; k \in K. \tag{2.12}$$

A função objetivo (2.1) modela os custos totais de viagem relacionados às rotas. As restrições (2.2) e (2.3) garantem que cada nó é servido por exatamente um veículo e que a coleta e a entrega são feitas pelo mesmo veículo. As restrições (2.4)-(2.6) garantem que cada rota se inicia no depósito inicial e termina no depósito final, e que o mesmo veículo que chega em um nó sai deste mesmo nó. A consistência do tempo e carga do veículo é garantida pelas restrições (2.7) e (2.8), respectivamente. Em (2.8) não é necessária a igualdade, pois as variáveis livres que assumirem valores na função objetivo não interferem na solução do problema. As restrições (2.9) garantem que, para cada pedido i, o nó de coleta é visitado antes do nó de entrega, enquanto que (2.10) e (2.11) impõem que as janelas de tempo e a capacidade sejam respeitadas, respectivamente. A última restrição (2.12) garante o domínio das variáveis de decisão.

Esta formulação não é linear devido às restrições (2.7) e (2.8). Introduzindo uma

constante suficientemente grande, $M > 0$, estas podem ser linearizadas como se segue:

$$B_{jk} \geq B_{ik} + d_i + t_{ij} - M\left(1 - x_{ijk}\right), \forall i \in N, j \in N, k \in K, \tag{2.13}$$

$$Q_{jk} \geq Q_{ik} + q_j - M\left(1 - x_{ijk}\right), \forall i \in N, j \in N, k \in K. \tag{2.14}$$

Esta formulação de 3-índices possui restrições da ordem polinomial, enquanto que as formulações de 2-índices apresentadas mais à frente possuem restrições da ordem exponencial. Porém, a formulação de 3-índices possui a relaxação linear fraca em relação ao modelo de 2-índices, isto é, o limitante inferior fornecido pela relaxação linear, em geral, é distante do valor ótimo do problema original. Além disso, este modelo possui um número maior de variáveis que o modelo de 2-índices.

Ropke et al. (2007) apresentaram duas formulações de 2-índices para o PDPTW, as quais são discutidas a seguir. Estas duas formulações são resolvidas utilizando-se algoritmos *branch-and-cut* também propostos pelos autores, já que as restrições são de ordem exponencial sendo, portanto, inviável a enumeração total das restrições.

Para impor os pares de coleta-entrega e as relações de precedência, é conveniente definir o conjunto $\mathcal{S}$ que é caracterizado por conter todos os subconjuntos de nós $S \subseteq N$, tal que $0 \in S$, $2n + 1 \notin S$, e existe pelo menos um pedido i tal que $i \notin S$ e $n + i \in S$. Para cada subconjunto $S \subseteq N$, definimos o complementar $N \backslash S$. Note que a definição do conjunto $S \in \mathcal{S}$ em (2.18) impõe que o depósito inicial 0 esteja contido em S e que pelo menos uma entrega deva pertencer ao conjunto S em que a coleta não esteja em S. A restrição (2.18) faz com que a rota se inicie em S, porque o depósito pertence a S e, em algum momento, saia do conjunto para coletar aquela demanda que não pertence a S, digamos j, e retorne ao conjunto para entregar a demanda de j, o nó $j + n$. Isto está ilustrado na Figura 1, em que $S = \{0,j + n,i\}$ e $j \notin S$. Observe que $|S| = 3$ e $\sum_{i,j \in S} x_{ij} = 1$.

Para cada arco $(i,j) \in A$, seja x_{ij} uma variável binária que é igual a 1 se e somente se um veículo viaja imediatamente do nó i para o nó j. Para cada nó $i \in P \cup D$, seja B_i o tempo de início do serviço em i, e Q_i a carga do veículo após visitar o nó i. Sendo assim, uma formulação de 2-índices para o PDPTW é dada por:

$$(PDPTW2) \qquad \text{Min} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{2.15}$$

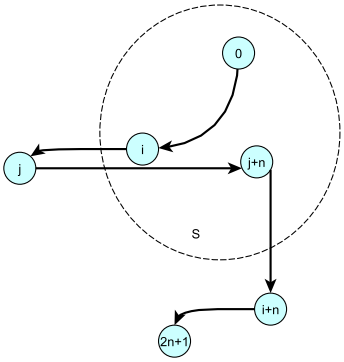

Figura 1: Exemplo de um conjunto $S \in \mathcal{S}$.

s.a

$$\sum_{i \in N} x_{ij} = 1 \qquad \forall j \in P \cup D \tag{2.16}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in P \cup D \tag{2.17}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 2 \qquad \forall S \in \mathcal{S} \tag{2.18}$$

$$B_j \geq (B_i + d_i + t_{ij})\, x_{ij} \qquad \forall i \in N; j \in N \tag{2.19}$$

$$Q_j \geq (Q_i + q_j)\, x_{ij} \qquad \forall i \in N; j \in N \tag{2.20}$$

$$e_i \leq B_i \leq l_i \qquad \forall i \in N \tag{2.21}$$

$$\max\{0,q_i\} \leq Q_i \leq \min\{Cap,Cap + q_i\} \qquad \forall i \in N \tag{2.22}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in N; j \in N. \tag{2.23}$$

A função objetivo (2.15) modela os custos totais relacionados às viagens. As restrições (2.16) e (2.17) impõem que cada nó seja visitado uma única vez. As restrições (2.18) são as restrições de precedência que garantem que a coleta de um dado nó i seja feita antes da entrega deste nó $(n + i)$; e além disso, ambos os nós de coleta e entrega respetivos devem ser visitados pelo mesmo veículo. A consistência no tempo de viagem e da carga do veículo são garantidas pelas restrições (2.19) e (2.20), respectivamente. As janelas de tempo e a garantia de que a capacidade do veículo seja respeitada são impostas pelas restrições (2.21) e (2.22), respectivamente. As restrições (2.19) e (2.21) garantem que não existem sub-rotas na solução. As restrições (2.23) garantem o domínio das variáveis de decisão.

A formulação (2.15)-(2.23) não é linear devido às restrições (2.19) e (2.20). Introdu-

zindo uma constante suficientemente grande $M > 0$, as restrições podem ser linearizadas como se segue:

$$B_j \geq B_i + d_i + t_{ij} - M\left(1 - x_{ij}\right), \forall i \in N, j \in N, \tag{2.24}$$

$$Q_j \geq Q_i + q_j - M\left(1 - x_{ij}\right), \forall i \in N, j \in N. \tag{2.25}$$

Ropke et al. (2007) também propõem uma segunda formulação de 2-índices, na qual tem-se um conjunto $\mathcal{R}$ de todas as rotas R infactíveis em relação às janelas de tempo. Seja $A(R)$ e $N(R)$ o conjunto de arcos e o conjunto de nós do conjunto $R \in \mathcal{R}$. Além disso, para qualquer subconjunto $S \subseteq P \cup D$, definimos $q(S) = \sum_{i \in S} q_i$. Dado o conjunto $S \subseteq N \backslash \{0\}$, denotamos por $r(S)$ o número mínimo de veículos necessários para servir os nós de S, que pode ser obtido pela solução ótima do problema de empacotamento (*Bin Packing Problem* - BPP) (MARTELLO; TOTH, 1990). Frequentemente, $r(S)$ é substituído por um limitante inferior trivial do BPP, dado por $\lceil q(S)/Cap \rceil$. Um limitante inferior para o número de veículos que devem entrar e sair de S, respeitando-se a capacidade Cap de cada veículo, é dado por:

$$\max\left\{1, \lceil |q(S)|/Cap \rceil\right\}. \tag{2.26}$$

Usando a notação definida, o segundo modelo de 2-índices para o PDPTW é dado por:

$$(PDPTW3) \qquad \text{Min} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{2.27}$$

s.a

$$\sum_{i \in N} x_{ij} = 1 \qquad \forall j \in P \cup D \tag{2.28}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in P \cup D \tag{2.29}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 2 \qquad \forall S \in \mathcal{S} \tag{2.30}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - \max\left\{1, \lceil |q(S)|/Cap \rceil\right\} \qquad \forall S \subseteq N \backslash \{0,2n+1\}; |S| \geq 2 \tag{2.31}$$

$$\sum_{(i,j) \in A(R)} x_{ij} \leq |A(R)| - 1 \qquad \forall R \in \mathcal{R} \tag{2.32}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in N; j \in N. \tag{2.33}$$

As restrições (2.28)-(2.30) e (2.33) são as mesmas do modelo PDPTW2 definido anteriormente. As restrições (2.31) impõem que a capacidade do veículo seja respeitada, enquanto que as restrições (2.32) eliminam qualquer caminho que seja infactível de acordo com as janelas de tempo. Observe que o conjunto S em (2.31) é qualquer subconjunto de $N \setminus \{0,2n+1\}$ com cardinalidade maior ou igual a 2, enquanto que o conjunto S em (2.30) deve satisfazer as condições descritas anteriormente para $\mathcal{S}$.

Há algumas variações e extensões do modelo de 2-índices relevantes na prática e que tem interesse para essa tese, conforme apresentado na sequência:

1. Considerar vários depósitos, por exemplo, um para cada veículo. Nesse caso, os modelos de 2-índices apresentados podem ser adaptados, fazendo-se cópias dos nós 0 e $2n+1$. Isto pode ser útil no caso em que os veículos não se encontrem no mesmo ponto no início do horizonte de tempo. Este tipo de modelagem é utilizado no modelo apresentado para o problema do estudo de caso, ou seja, é utilizado no decorrer deste trabalho;

2. Permitir que alguns veículos não sejam utilizados. Nesse caso, podemos modelar as restrições para que certos veículos viajem apenas do depósito inicial 0 para o depósito final $2n+1$, sem realizar visitas a nenhum nó. Esta adaptação também vai ser útil ao problema do estudo de caso, já que alguns veículos podem não estar disponíveis no início do roteamento;

3. No caso em que cada veículo possui uma capacidade diferente dada por $Cap_k, k \in K$, com $Cap_k \leq Cap_{max}$ (Cap_{max} dado pela capacidade do maior veículo), o modelo de 2-índices pode ser adaptado. Assim, no depósito inicial todos os veículos possuem capacidade igual a Cap_{max} e, a primeira visita é feita a um nó fictício, em que cada veículo coleta a quantidade $Cap_{max} - Cap_k, \ \forall k \in K$. Desta maneira, depois da primeira visita, cada veículo possui uma capacidade diferente. Esta quantidade fictícia é entregue somente no último nó da rota do veículo específico, este também é um nó fictício. Este procedimento serve para representar problemas com frota heterogênea, em que a diferença da frota se limita nas capacidades diferentes dos veículos. Tem-se que esta abordagem também é utilizada no decorrer deste trabalho, mais especificamente, na proposta do método *branch-and-cut*.

Os modelos de 2-índices são utilizados em muitas versões básicas do PDP e algu-

mas de suas variantes, mas geralmente é inadequado para versões mais completas. Além disso, somente são apropriados para modelar situações cuja função objetivo não dependa da sequência dos nós em uma dada rota, ou do tipo de veículo utilizado. Uma possibilidade de lidar com estas limitações da modelagem de 2-índices é indicar explicitamente os veículos, resultando em formulações com variáveis de 3-índices, as quais são mais flexíveis para incorporar novas restrições, porém o número de variáveis pode aumentar consideravelmente.

O modelo PDPTW2 é mais flexível que o modelo PDPTW3, pois pode ser considerada uma função objetivo mais geral, envolvendo as variáveis de tempo e de quantidade de carga do veículo. Já o modelo PDPTW3 possui menos variáveis e pode ter um tempo computacional inferior ao do modelo PDPTW2, conforme observado por Ropke et al. (2007). Os autores compararam o desempenho dos modelos PDPTW2 e PDPDTW3 com o PDPTW1. Os experimentos mostraram que as três formulações são competitivas, entretanto as formulações de 2-índices têm melhores resultados, principalmente a formulação PDPTW3. Além disso, os limitantes inferiores das formulações PDPTW2 e PDPTW3 são melhores que os limitantes da formulação de 3-índices (PDPTW1).

Uma quarta formulação proposta na literatura para o PDPTW e baseada na formulação de particionamento de conjuntos é dada a seguir. Seja $\mathcal{H}$ o conjunto de todas as rotas factíveis, ou seja, rotas que satisfazem a capacidade do veículo, janelas de tempo e precedência. Para cada rota $j \in \mathcal{H}$, seja c_j o custo da rota e a_{ij} a constante que indica o número de vezes que o nó $i \in P$ é visitado pela rota j. Além disso, y_j é uma variável binária que assume valor 1 se a rota $j \in \mathcal{H}$ é utilizada na solução ótima. Sendo assim, a modelagem de particionamento de conjuntos que Ropke e Cordeau (2009) propõem para o PDPTW é dada por:

$$(PDPTW4) \qquad \text{Min} \sum_{j \in \mathcal{H}} c_j y_j \tag{2.34}$$

s.a

$$\sum_{j \in \mathcal{H}} a_{ij} y_j = 1 \qquad \forall i \in P \tag{2.35}$$

$$y_j \in \{0,1\} \qquad \forall j \in \mathcal{H}. \tag{2.36}$$

A função objetivo (2.34) modela os custos de todas as rotas que forem escolhidas na

solução ótima. As restrições (2.35) garantem que cada nó é visitado uma única vez e as restrições (2.36) impõem o domínio da variável y.

Como o conjunto $\mathcal{H}$ possui cardinalidade tipicamente grande, é difícil representar o modelo PDPTW4 explicitamente. Então, normalmente utiliza-se o método de geração de colunas que permite considerar apenas um subconjunto de rotas $\overline{\mathcal{H}} \subseteq \mathcal{H}$. Neste contexto, o problema (2.34)-(2.36) é chamado de problema mestre e o problema que considera apenas um subconjunto de colunas é denominado problema mestre restrito. As colunas são geradas sempre que necessário e isto pode ser feito resolvendo subproblemas que oferecem a coluna mais atrativa para entrar no problema mestre restrito. Para problemas de roteamento de veículos, o subproblema é definido por um problema de caminho mínimo com restrições adicionais. Ropke e Cordeau (2009) propõem dois tipos de subproblemas: o primeiro baseado no ESPPTWCPD (*elementary shortest path problem with time windows, capacity, and pickup and delivery*) e o segundo baseado no SPPTWCPD (*shortest path problem with time windows, capacity, and pickup an delivery*) e comparam o desempenho de ambos no método. Os resultados mostram que o método *branch-and-cut-and-price* proposto obteve melhores resultados, comparado aos métodos *branch-and-cut* da literatura.

O que podemos concluir com esses quatro modelos apresentados é que cada um possui algumas vantagens e desvantagens de aplicação, todos eles com bom potencial para servir de ponto de partida para a modelagem e resolução do problema abordado nesta tese. Um dos nossos objetivos é analisar o desempenho de extensões dos modelos PDPTW1, PDPTW2, PDPTW3 e PDPTW4 representados para o nosso caso específico e compará-los utilizando exemplares da literatura e exemplares do estudo de caso.

O conteúdo mais importante e que serve para os próximos capítulos desta tese é a notação matemática, juntamente com os modelos apresentados. O modelo PDPTW1 é utilizado como base no Capítulo 4 para modelar o problema de coleta e entrega na indústria petrolífera. Os modelos PDPTW1, PDPTW2 e PDPTW3 são utilizados no Capítulo 5 para representar os modelos que servem de base para os métodos do tipo *branch-and-cut* propostos. Já o PDPTW4 é utilizado para representar o modelo que servirá de base para o método *branch-and-price*.

3 Revisão de Métodos Exatos

Neste capítulo, são brevemente revisados os métodos *branch-and-bound*, *branch-and-cut* e *branch-and-price*, bem como suas características e definições. Ao final, descrevemos em detalhes algumas desigualdades válidas utilizadas nos métodos e que são propostas na literatura. Inicia-se pelo método *branch-and-bound*, descrito resumidamente a seguir.

3.1 Método *branch-and-bound*

Uma maneira de encontrar uma solução ótima de um problema de otimização é utilizar a enumeração implícita, que consiste em reduzir o espaço de busca de modo a descartar subconjuntos de soluções que não contém a solução ótima. Estes subconjuntos são descartados utilizando os limitantes superior e inferior para o valor da solução ótima. Esta é a ideia do algoritmo *branch-and-bound*, utilizado para encontrar soluções ótimas para vários problemas de otimização, especialmente em otimização combinatória.

A ideia básica deste método é relaxar o problema, por exemplo, usando-se a relaxação linear, em que se coloca as variáveis inteiras como contínuas. Esta é a relaxação mais comumente utilizada, mas outras podem ser usadas como, por exemplo, relaxação combinatória (esta é obtida a partir da programação linear inteira em que algumas restrições são retiradas do modelo, por exemplo, para o CVRP as restrições que impõem eliminação de sub-rota e que a capacidade do veículo seja respeitada são as retiradas). A partir daí divide-se este problema principal em outros subproblemas. Tipicamente tem-se 2 subproblemas obtidos ao escolher uma variável fracionária, digamos x_i, que tenha assumido valor fracionário $\bar{x}_i$ na solução ótima da relaxação linear. Em um dos subproblemas, impõe-se $x_i \leq \lfloor \bar{x}_i \rfloor$ (maior inteiro menor ou igual a $\bar{x}_i$) enquanto que no outro subproblema impõe-se $x_i \geq \lceil \bar{x}_i \rceil$ (menor inteiro maior ou igual a $\bar{x}_i$). Em particular, em problemas em que a variável é binária, em um dos subproblemas, x_i é fixado em 1 e, no outro, em 0. Então, resolve-se o problema relaxado com uma das fixações e a outra fixação fica armazenada em um conjunto de nós ativos, que devem ser testados. Depois de um subproblema ser

resolvido, este pode ser ramificado novamente ou o nó pode ser eliminado da árvore, isto é, este nó e nenhum de seus filhos contém a solução ótima. A eliminação dos nós é feita de acordo com algumas regras que serão discutidas mais a frente nesta seção e resolve-se estes passos até que a solução ótima seja encontrada.

A relaxação linear de alguns problemas é dita ser fraca, como já mencionado, quando seu valor ótimo está muito distante do valor ótimo do problema original. Sendo assim, existem estudos que abordam este tema, analisando outras formas de relaxação para que esta se torne mais forte. O primeiro estudo a aplicar o algoritmo *branch-and-bound* para o CVRP foi Laporte et al. (1986), que investiga e testa outros tipos de relaxações diferentes da linear. Para outras informações e detalhes, ver Toth e Vigo (2002c). Uma revisão mais atual que abrange também outros métodos exatos é encontrada em Laporte (2009).

3.1.1 Algoritmo *branch-and-bound*

Antes de descrever o algoritmo *branch-and-bound* para problemas de minimização, definimos algumas notações: PL^i é a relaxação linear do nó i e $F(PL^i)$ a região factível desta relaxação. Seja $\overline{z}^i$ o valor ótimo do problema PL^i (limitante inferior de z), x^* a melhor solução encontrada até o momento e z^* o valor de x^*. O método *branch-and-bound* é descrito no Algoritmo 1, considerando-se que os limitantes são obtidos resolvendo-se as relaxações lineares PL^i. Um nó ainda não eliminado por nenhuma das regras de eliminação é dito ser um nó ativo. Todos os nós ativos são armazenados em uma lista L.

Há diversas regras para processar os nós na lista L de nós ativos. A mais simples é a regra *last-in, first-out* em que o último nó a entrar na lista é o primeiro a sair. Assim, determina-se uma busca em profundidade pela solução ótima na árvore de busca. Esta regra de processamento pode produzir uma árvore com muitos nós, o que pode ser inviável. Entretanto, a literatura diz que encontrar a solução ótima é mais fácil em níveis mais profundos da árvore, e além disso, nesta regra a lista L de nós ativos não é muito grande. Outra regra para processar os nós ativos é analisar primeiramente os nós com maior limitante superior para problemas de maximização ou escolher os nós com menor limitante inferior, para problemas de minimização. Esta árvore, em geral, possui menos nós no total comparada à outra regra. Porém, a desvantagem é que o número de nós ativos pode ser maior.

Algoritmo 1: Algoritmo *branch-and-bound.*

```
1  enquanto Existirem nós na lista L faça
2      i = 0, PL^0 = RL(P);
3      Resolver a relaxação PL^i;
4      se Achou solução ótima do problema relaxado então
5          se z̄^i < z* então
6              Obter Índice de Ramificação;
7              se não achou índice de ramificação (ou seja, é solução inteira) então
8                  Atualiza z*;
9                  Atualiza x*;
10                 Descartar nó i por otimalidade;
11             senão
12                 Ramifica;
13         senão
14             Descarta nó i por qualidade;
15     senão
16         Descartar nó i por infactibilidade;
17     enquanto Existirem nós na lista L e z̄^i ≥ z* faça
18         Descartar nó i por qualidade;
19         Definir i como o próximo nó de L;
```

3.1.2 Estratégias de ramificação e corte dos nós

Existem várias estratégias utilizadas para ramificar os nós da árvore de busca. Para problemas de roteamento e programação de veículos, geralmente resolve-se a relaxação linear e, a partir da solução ótima da relaxação, ramifica-se na variável (digamos x_{ij}) com valor mais próximo de 0,5. Se houver duas variáveis com o mesmo valor, utiliza-se um segundo critério, por exemplo, escolhe-se aquela que tenha maior custo na função objetivo. Então, um subproblema terá esta variável com valor fixado em 1 ($x_{ij} = 1$) e outro subproblema esta variável terá valor fixado em 0 ($x_{ij} = 0$) (PADBERG; RINALDI, 1991). Entretanto, existem outras regras de ramificação que serão enumeradas a seguir:

1. A ramificação é feita na variável que estende um caminho já existente e com maior demanda (NADDEF; RINALDI, 2002);

2. A variável mais próxima de 0,75 é ramificada; se tiverem duas escolhe-se a com maior custo (NADDEF; RINALDI, 2002);

3. Tem-se o conjunto de todas as variáveis que podem ser ramificadas, então escolhe-se uma candidata (x_{ij}) e resolve-se o problema linear para $x_{ij} = 1$ e, depois, para

$x_{ij} = 0$. A fixação escolhida é aquela que minimiza a função objetivo (APPLEGATE et al., 1994);

4. Ramificar usando algum subconjunto $S \subseteq N$ tal que $\left\lceil \frac{q(S)}{Cap} \right\rceil = t$, sendo t um número inteiro. O número de arcos que entram e saem deste subconjunto, ou seja, o número de arcos em que um nó está em S e o outro nó em $N \backslash S$ seja próximo de $2t + 1$. Então, um subproblema é fixado em $2t$ e o outro em $2t + 2$. Isto é chamado de ramificação nas inequações e a melhor situação é quando $t = 1$. (AUGERAT et al., 1995).

Para a eliminação de nós na árvore de busca existem três regras que são descritas a seguir. Sendo assim, o problema referente ao nó i pode ser eliminado da árvore de busca se satisfizer pelo menos uma das seguintes condições:

1. O problema referente ao nó i é eliminado por infactibilidade se $F(PL^i) = \varnothing$;
2. O nó i é eliminado por qualidade se $\overline{z}^i \geq z^*$;
3. Se a solução ótima de PL^i for inteira tal que $\overline{z}^i < z^*$, então o nó i é eliminado por otimalidade.

3.2 Método *branch-and-cut*

O método *branch-and-cut* é uma modificação do algoritmo de *branch-and-bound* que utiliza planos de corte com o intuito de melhorar a qualidade da relaxação linear. Os planos de corte são restrições adicionais que buscam aproximar a envoltória convexa da região factível de um problema inteiro (BALAS et al., 1996). Em cada nó, depois de resolvida a relaxação linear, o algoritmo de planos de corte adiciona uma ou mais restrições violadas pela solução ótima da relaxação linear, mas que são satisfeitas por qualquer solução ótima do problema. Estas restrições recebem o nome de desigualdades válidas ou cortes (PADBERG; RINALDI, 1991; CAPRARA; FISCHETTI, 1997).

Seja $\overline{z}^i$ o valor da solução ótima da relaxação linear do PL^i. Define-se por $PL(\infty)$ a relaxação linear com todas as desigualdades válidas possíveis. Ao invés de resolvermos este problema, vamos resolver um problema denominado $PL(h)$, para $h \geq 0$, que consiste num subconjunto de restrições do $PL(\infty)$. Se a solução do $PL(h)$ for inteira e factível, então esta solução é ótima. Caso contrário, assume-se que temos uma caixa preta que contém alguma inequação de $PL(\infty)$ que não está em $PL(h)$ e que é violada, e esta

é adicionada ao problema $PL\,(h+1)$. Esta caixa preta é denominada de algoritmo de separação e as inequações violadas são determinadas por esse algoritmo (TOTH; VIGO, 2002). A cada iteração do método de planos de cortes, novas desigualdades violadas são encontradas e acrescidas ao problema, até que a solução ótima seja encontrada ou que algum critério seja satisfeito, como violação dos cortes obtidos abaixo de um certo limiar ou número máximo de iterações atingido, dentre outros. Se não tivermos a solução ótima inteira, temos que ramificar, criando-se outros dois subproblemas, impondo limitantes a alguma variável, como é descrito para o método *branch-and-bound* na Seção 3.1.

Os planos de corte adicionados à relaxação, podem ser de dois tipos: i) desigualdades válidas, que têm o propósito de melhorar a qualidade do limitante obtido, podendo ser cortes de propósito geral, como os de *Chvátal-Gomory* (WOLSEY, 1998); ii) restrições que foram retiradas da formulação devido ao uso da relaxação combinatória como, por exemplo, as restrições (2.18) e (2.32) das formulações PDPTW2 e PDPTW3, respectivamente. Estas restrições são inicialmente descartadas e, iterativamente, adicionadas ao problema. No método *branch-and-cut* o limitante produzido em cada nó é, em geral, melhor que no método *branch-and-bound*, e isto se deve às desigualdades válidas que são acrescentadas em cada iteração. Os cortes produzidos através dos algoritmos de separação produzem uma perturbação na solução fracionária, levando a limitantes melhores.

3.2.1 Algoritmo *branch-and-cut*

O Algoritmo 2 representa o algoritmo de planos de corte. O algoritmo *branch-and-cut* é dado substituindo-se a linha 3 do Algoritmo 1 pelo Algoritmo 2. É semelhante ao Algoritmo 1 exceto que em cada nó há a inclusão de cortes, ou seja, de desigualdades válidas.

Os primeiros trabalhos a usarem o método *branch-and-cut* são aplicados na resolução do TSP (GRÖTSCHEL; PADBERG, 1979; FLEISCHMANN, 1988; PADBERG; RINALDI, 1991; BALAS et al., 1995; ASCHEUER et al., 2000, 2001). Com isto, temos que a maioria das desigualdades válidas são geradas para o TSP, e algumas destas são também válidas para outros problemas relacionados, como o VRP e o PDP. Outras desigualdades são desenvolvidas especificamente para o VRP e PDP (LYSGAARD, 2006; CAPRARA; FISCHETTI, 1997). O sucesso de um método *branch-and-cut* está também associado à estrutura do problema, que diz quais são as desigualdades válidas que aproximam do envoltório convexo do problema. A seguir, apresentamos alguns trabalhos importantes que desenvolveram o método *branch-and-cut* para resolver o VRP e suas variantes.

Algoritmo 2: Algoritmo de planos de corte.

```
1  para t = 1,...,T faça
2      Defina o PL;
3      Resolva o PL;
4      Seja x^t a solução ótima do PL;
5      se x^t ∈ ℤ^n então
6          Pare;
7      senão
8          Encontre uma desigualdade válida;
9      se Encontrou uma desigualdade válida então
10         Acrescente a desigualdade ao PL;
11         Incremente t e repita o procedimento;
12     senão
13         Pare;
```

Em Lysgaard et al. (2004) é apresentado um novo algoritmo *branch-and-cut* para o problema de roteamento de veículos capacitado. Para este caso, a frota é homogênea e o objetivo é minimizar os custos das viagens. São apresentados os seguintes planos de corte: *capacity inequalities*, *framed capacity*, *generalized capacity*, *strengthened comb*, *multistar*, *extended hypotour inequalities* e o clássico corte de Gomory. Os resultados mostram que o método é competitivo com outros métodos da literatura e, além disso, é possível provar a otimalidade de três exemplares pela primeira vez.

Em Yaman (2006) é estudado o problema de roteamento de veículos com frota heterogênea. O autor melhora o limitante inferior através das inequações válidas do tipo *Covering type inequalities*, eliminação de sub-rota e também as chamadas *multistar*. Para as duas últimas o autor generaliza as inequações e, com isto, desenvolve seis diferentes formulações. O problema em que os veículos não são obrigados a retornar ao depósito é tratado em Letchford et al. (2007). Os autores destacam que, por mais que seja uma pequena variação do VRP clássico, os planos de corte precisaram ser modificados e assim, um algoritmo *branch-and-cut* específico é apresentado. Outra variação do VRP clássico é o problema de roteamento e localização de depósitos que é estudado por Belenguer et al. (2011), para o qual os autores também propõem um algoritmo *branch-and-cut* para sua resolução.

No contexto marítimo podemos citar o trabalho de Moccia et al. (2006), em que o problema da programação de guindastes de um porto é modelado e resolvido por um algoritmo *branch-and-cut*, em que são incorporadas várias famílias de desigualdades válidas, inclusive restrições de precedência. Ainda na linha de transporte marítimo, Reinhardt e

Pisinger (2012) modelam o problema de rede e atribuição de frota como um modelo de programação inteira mista e o resolvem por um algoritmo *branch-and-cut*. Outros trabalhos e detalhamentos sobre os métodos podem ser encontrados em Naddef e Rinaldi (2002), Berbeglia et al. (2007), Parragh et al. (2008) e Parragh et al. (2008b).

Nesta seção damos maior ênfase aos algoritmos *branch-and-cut* propostos para o VRP e para o TSP. Para o PDP, os trabalhos com métodos exatos são citados na Seção 2.2 que é específico sobre o problema de coleta e entrega. Ao final deste capítulo são apresentadas algumas desigualdades válidas que são utilizadas no desenvolvimento dos métodos exatos propostos, especificamente para o problema de coleta e entrega com janelas de tempo na indústria petrolífera no Capítulo 4.

3.3 Método *branch-and-price*

O método *branch-and-price* utiliza a estratégia *branch-and-bound* combinada com a obtenção de limitantes inferiores utilizando geração de colunas. Sendo assim, o *branch-and-price* busca melhorar este limitante para tornar o *branch-and-bound* mais eficaz. Em geral, um método de decomposição é utilizado para obter uma relaxação que ofereça melhores limitantes como, por exemplo, a decomposição de *Dantzig-Wolfe* (DDW). A DDW tem sido bastante utilizada para problemas de programação inteira e obtido bons resultados em sua implementação (VANDERBECK, 2000). Revisões sobre o método de geração de colunas e método *branch-and-price* podem ser encontrados em Lübbecke e Desrosiers (2005) e Desrosiers e Lübbecke (2010), respectivamente.

A DDW envolve a resolução de diversos problemas menores e mais fáceis de serem resolvidos ao invés de resolver o problema original, que pode possuir muitas variáveis e restrições. Em geral, a decomposição explora alguma estrutura especial da matriz de coeficientes. Considere o problema de programação inteira com a seguinte estrutura:

$$\text{Min} \qquad c^T x \tag{3.1}$$

s.a

$$Ax = b \tag{3.2}$$

$$Dx = d \tag{3.3}$$

$$x \in \mathbb{Z}_+^n. \tag{3.4}$$

em que A e D são matrizes e b e d são vetores.

A estrutura adequada à DDW é dada por uma das opções a seguir: 1) as restrições (3.2) são complicadoras e difíceis de serem resolvidas e a resolução do problema fica restrito às restrições (3.3), que são mais fáceis; 2) as restrições (3.3) têm uma estrutura de blocos diagonal e as restrições (3.2) acoplam os blocos; 3) considerar as restrições (3.2) e (3.3) separadamente é mais tratável, mas quando juntas tornam o sistema mais difícil. Vamos aplicar a DDW ao modelo (3.1)-(3.4) utilizando a técnica de discretização, conforme descrito por Vanderbeck (2000).

Seja $P = \left\{x \in \mathbb{Z}_+^n | Dx = d\right\} \neq \varnothing$, então existe um conjunto finito de pontos discretos (factíveis) de $\mathcal{X}$ $\left(\{p_k\}_{k\in K} \subseteq \mathcal{X}\right)$ tal que

$$\mathcal{X} = \left\{ x \in \mathbb{Z}_+^n | x = \sum_{k\in K} p_k \lambda_k, \sum_{k\in K} \lambda_k = 1, \lambda_k = \{0,1\} \right\}. \tag{3.5}$$

Sendo assim, substituindo x de (3.5) no problema (3.1)-(3.4) obtemos o problema mestre da DDW:

$$z_{PM} = \text{Min} \sum_{k\in K} \left(c^T p_k\right) \lambda_k \tag{3.6}$$

s.a

$$\sum_{k\in K} \left(A p_k\right) \lambda_k = b \tag{3.7}$$

$$\sum_{k\in K} \lambda_k = 1 \tag{3.8}$$

$$\lambda_k \in \{0,1\} \qquad\qquad \forall k \in K. \tag{3.9}$$

em que o vetor b tem dimensão $(m_1 \times n)$.

A relaxação linear do problema (3.6)-(3.9) se dá pela relaxação da integralidade da

variável λ, ou seja, substituindo a restrição (3.9) por

$$\lambda_k \geq 0, \ \ \forall k \in K. \tag{3.10}$$

Seja z_{PML} o valor ótimo do problema (3.6)-(3.8) e (3.10), ou seja, a relaxação linear do problema mestre (PML), e seja z_{PL} o valor ótimo da relaxação linear de (3.1)-(3.4). Então, $z_{PL} \leq z_{PML}$, pois uma solução factível da relaxação (3.1)-(3.4) que não é combinação convexa dos pontos discretos de $\mathcal{X}$, não é factível em (3.7). Considere agora as variáveis duais $\{\pi_i\}_{i=1}^{m_1}$ associadas às restrições (3.7), e seja μ a variável dual associada à restrição (3.8). O problema mestre linear restrito (PMLR) é dado por:

$$z_{PM} = \text{Min} \sum_{k \in \Lambda} \left(c^T p_k\right) \lambda_k \tag{3.11}$$

s.a

$$\sum_{k \in \Lambda} \left(A p_k\right) \lambda_k = b \tag{3.12}$$

$$\sum_{k \in \Lambda} \lambda_k = 1 \tag{3.13}$$

$$\lambda_k \geq 0, \qquad \forall k \in \Lambda. \tag{3.14}$$

em que $\Lambda \subset K$ contém uma base para o PMLR.

Então, para verificar a otimalidade da solução do PML, um subproblema chamado *pricing* é resolvido para encontrar colunas que possam reduzir o valor da função objetivo (para um problema de minimização). Trata-se de encontrar uma coluna que tenha um custo reduzido negativo. Sendo assim, o subproblema é dado por:

$$\eta = \min \left\{ \left(c^k - \pi \left(A^k\right)^T\right)^T x^k - \mu : x^k \in \mathcal{X} \right\}, \tag{3.15}$$

em que (π,μ) é a solução dual do PMLR.

O problema mestre (3.6)-(3.9) possui um número extremamente grande de variáveis, tipicamente exponencial em n. Então, o método de geração de colunas é muito utilizado para resolver este tipo de formulação. Este método é iterativo e inicia-se resolvendo o problema mestre restrito para obter a solução dual, e utiliza a solução dual para gerar uma ou mais colunas para serem adicionadas ao PMLR. Se $\eta \geq 0$, significa que a solução ótima

foi encontrada. Caso contrário, seja $\widetilde{x}$ a solução ótima do subproblema. Então, a coluna $\left(c^T\widetilde{x}, A\widetilde{x}, 1\right)^T$ é incluída no PMLR e, assim, o problema mestre restrito é reotimizado até que todos os custos reduzidos sejam maiores ou iguais a 0. A rotina completa do método de geração de colunas é dado pelo Algoritmo 3. Este algoritmo juntamente com o Algoritmo 1 resulta no método *branch-and-price*. O resultado do PML, ou seja, z_{PML} é tal que $z_{PML} \leq z$, isto é, um limitante inferior de z que é usado no método *branch-and-bound*.

O método *branch-and-cut-and-price* consiste em uma variação do método *branch-and-price* em que desigualdades válidas são utilizadas para melhorar os limitantes obtidos e tornar o método mais eficaz, assim como é feito no método *branch-and-cut*. A diferença para este último é que as desigualdades válidas são acrescidas ao problema mestre restrito. Assim, a revisão é feita comumente para os dois métodos.

Algoritmo 3: Algoritmo de Geração de Colunas.

```
1 Defina o Problema Mestre (PM);
2 Defina o Problema Mestre Restrito (PMR);
3 Encontre um conjunto inicial de colunas para o PMR;
4 Resolva o PMR linear;
5 Resolva o subproblema e encontre as variáveis duais ótimas;
6 se Custo relativo for positivo então
7     Pare;
8 senão
9     Adicione a coluna encontrada ao PMR e volte ao passo 4;
```

O primeiro método *branch-and-price* proposto para o PDPTW foi em Dumas et al. (1991), em que os autores consideraram a formulação de particionamento de conjuntos em que cada coluna corresponde à rota factível de um veículo e cada restrição está associada à uma requisição que deve ser satisfeita por exatamente um veículo. O subproblema corresponde a um problema de caminho mínimo.

Para o problema de roteamento e programação de veículos com frota heterogênea, janelas de tempo e múltiplos depósitos, Bettinelli et al. (2011) propõem um algoritmo *branch-and-cut-and-price*, em que os veículos possuem custos fixos e apenas diferem em sua capacidade. São estudadas diferentes técnicas de se gerar os planos de corte e as colunas e, assim, os resultados reportam estas combinações. Ainda, citando variantes do VRP, Baldacci et al. (2010) propõem um algoritmo para variações do VRP que podem ser modeladas como particionamento de conjuntos. Então, um algoritmo *column-and-cut* é proposto e, através dele, os autores descrevem como este pode ser estendido para as demais variantes, como CVRP, VRPTW, PDP, dentre outras.

Em Gutiérrez-Jarpa et al. (2010) os autores propõem um algoritmo *branch-and-price* para o VRP com entregas, coletas seletivas e janelas de tempo. Nesta variação, o conjunto de nós é a união dos nós de coleta e entrega, a frota é heterogênea e o objetivo é minimizar os custos relacionados às rotas. Já o VRPTW com tempo dependente da rota é resolvido por um algoritmo *branch-and-price* por Dabia et al. (2012). Outra variante que é resolvida por métodos exatos é o VRP com janelas de tempo do tipo *soft* e tempos de viagens estocásticos (TAS et al., 2013). Neste trabalho, um método *branch-and-price* é proposto e um procedimento de geração de colunas é utilizado, em que o problema mestre é modelado como particionamento de conjuntos e o subproblema (um para cada veículo) corresponde ao problema de caminho mínimo. Outras variações do VRP que são resolvidas por métodos do tipo *branch-and-price* podem ser encontradas em Archetti et al. (2011), Salani e Vacca (2011) e Munari e Gondzio (2013).

Em Pessoa et al. (2009) os autores apresentam um algoritmo robusto do tipo *branch-cut-and-price* para o problema de roteamento e programação de veículos, com frota heterogênea em que os veículos podem ter capacidades e custos distintos. Para a formulação do problema são acrescentados cortes válidos que são cruciais para garantir a consistência dos resultados do algoritmo. Os resultados computacionais mostraram que o método proposto é mais eficiente na resolução das instâncias comparado a outros trabalhos no mesmo contexto.

Bettinelli et al. (2014) os autores propuseram um algoritmo *branch-and-price* para o problema de coleta e entrega com frota heterogênea, múltiplos depósitos e janelas de tempo que podem ser violadas. Os autores detalham uma maneira de tratar dos subproblemas que será utilizada neste trabalho. Em Cherkesly et al. (2015a) os autores propõem modelos e dois algoritmos do tipo *branch-and-price-and-cut* para o problema de coleta e entrega com janelas de tempo e múltiplas pilhas. Este problema está inserido no contexto de transporte de materiais pesados ou perigosos e as pilhas são tratadas da forma *last-in-first-out* (LIFO). O primeiro algoritmo resolve o problema de caminho mínimo com o problema das pilhas e o segundo, incorpora este problema parte no caminho mínimo e parte através de cortes no problema mestre, quando rotas infactíveis são geradas a respeito das pilhas. Outros trabalhos envolvendo o problema de coleta e entrega e o método *branch-and-price* são Qu e Bard (2015), Cherkesly et al. (2015b).

No contexto de roteamento marítimo podemos citar Stålhane et al. (2012), em que os autores propõem um método do tipo *branch-and-cut-and-price* para o problema de coleta e entrega com carga fracionada. A frota é heterogênea e o objetivo é maximizar o lucro da

empresa. Os resultados computacionais mostram que, para alguns exemplares, o método proposto é mais eficaz que outros trabalhos propostos na literatura. Outro trabalho é o de Brønmo et al. (2010) em que um procedimento baseado na DDW é apresentado para o problema de programação dos navios com cargas flexíveis. Este problema é similar ao PDPTW e o método não é capaz de garantir todas as soluções ótimas. Outros trabalhos na área marítima que desenvolvem métodos exatos podem ser encontrados em Hwang et al. (2008) e Grønhaug et al. (2010).

3.4 Desigualdades válidas

As desigualdades descritas abaixo são apresentadas no trabalho de Ropke et al. (2007) para os modelos PDPTW2 e PDPTW3, descritos no capítulo anterior (Seção 2.2.2). A maioria destas desigualdades são utilizadas nos métodos do tipo *branch-and-cut* propostos no Capítulo 5 desta tese.

Eliminação de sub-rota:

Considere as inequações clássicas de eliminação de sub-rota propostas inicialmente para o TSP (FISHER; JAIKUMAR, 1981):

$$x(S) \leq |S| - 1, \tag{3.16}$$

para $S \subseteq P \cup D$ e $x(S) = \sum_{i,j \in S} x_{ij}$. Estas inequações são válidas para o problema clássico do VRP e também são válidas para o problema de coleta e entrega. Além disso, temos que estas inequações podem se tornar mais fortes se levarmos em conta que, dado um nó i, este possui apenas um sucessor e um antecessor. Ainda, especificamente para o contexto do problema de coleta e entrega, o nó i deve ser visitado antes do nó $n+i$ e pelo mesmo veículo. Cordeau (2006) primeiramente provou que as inequações a seguir propostas por Balas et al. (1995) para o problema assimétrico do TSP eram válidas também para o problema de coleta e entrega e janelas de tempo. Dado um conjunto $S \subseteq P \cup D$ e seu complementar $N \backslash S$, seja $\pi(S) = \{i \in P | n+i \in S\}$ o conjunto dos nós de coleta de S e $\sigma(S) = \{n+i \in D | i \in S\}$ o conjunto dos nós de entrega de S. Desta maneira, as seguintes inequações são válidas:

$$x(S) + \sum_{i \in N \backslash S \cap \sigma(S)} \sum_{j \in S} x_{ij} + \sum_{i \in N \backslash S \backslash \sigma(S)} \sum_{j \in S \cap \sigma(S)} x_{ij} \leq |S| - 1, S \subset P \cup D, \tag{3.17}$$

$$x(S) + \sum_{i \in S} \sum_{j \in N \setminus S \cap \pi(S)} x_{ij} + \sum_{i \in S \cap \pi(S)} \sum_{j \in N \setminus S \setminus \pi(S)} x_{ij} \leq |S| - 1, S \subset P \cup D. \tag{3.18}$$

Para a inequação (3.17), a parte que a torna mais forte em relação à inequação clássica de eliminação de sub-rota (3.16) é dada em duas parcelas pelos termos em somatórios do lado esquerdo da inequação. Isto melhora a inequação clássica, pois relaciona os nós de coleta que estão em S com as respectivas entregas que estão em $N \setminus S$. Além disso, o lado direito da inequação continua o mesmo, sendo que acrescenta-se mais arcos ao lado esquerdo da inequação e, assim, a torna mais apertada. Para a inequação (3.18) temos uma situação análoga.

Um exemplo descrito em Cordeau (2006) para a inequação (3.17) é considerar o conjunto de dois nós $S = \{i,j\} \subseteq P$. Sendo assim, temos que $\pi(S) = \varnothing$, pois i e j são nós de coleta e, portanto, não têm predecessores de coleta; e $\sigma(S) = \{n+i,n+j\}$. Na segunda parcela da inequação não consideramos os arcos $x_{n+i,i}$ e $x_{n+j,j}$, pois estes podem ser removidos previamente no pré-processamento. Desta maneira, temos a seguinte situação:

- A primeira parcela da inequação é $x(S) = x_{ij} + x_{ji}$;
- A segunda parcela corresponde a $\sum_{i' \in N \setminus S \cap \sigma(S)} \sum_{j' \in S} x_{i'j'}$ com $i' = \{n+i,n+j\}$ e $j' = \{i,j\}$, ou seja, $x_{n+i,j} + x_{n+j,i}$;
- A terceira parcela da inequação é dada por $\sum_{i' \in N \setminus S \setminus \sigma(S)} \sum_{j' \in S \cap \sigma(S)} x_{i'j'}$ e não há arcos nesta terceira parcela;
- No lado direito da inequação (3.17) temos que $|S| = 2$;
- Resultando na seguinte inequação: $x_{ij} + x_{ji} + x_{n+i,j} + x_{n+j,i} \leq 2 - 1 = 1$.

A inequação resultante está ilustrada na Figura 2. Os arcos pontilhados correspondem aos arcos que são melhorados na inequação, ou seja, os arcos que não fazem parte da eliminação de sub-rota clássica (3.16). A Figura 2 mostra que, para satisfazer a inequação (3.17), podemos ter apenas um dos quatro arcos desenhados na solução ótima. Isto significa que, além dos arcos tradicionais da inequação da eliminação de sub-rota clássica (que são os arcos cheios - não pontilhados), temos também os arcos pontilhados, ou seja, a inequação está mais forte.

Um outro exemplo dado em Cordeau (2006) é para a inequação (3.18). Considere $S = \{n+i,n+j\} \subset D$. Temos que $\pi(S) = \{i,j\}$ e $\sigma(S) = \varnothing$, pois $n+i$ e $n+j$ são nós

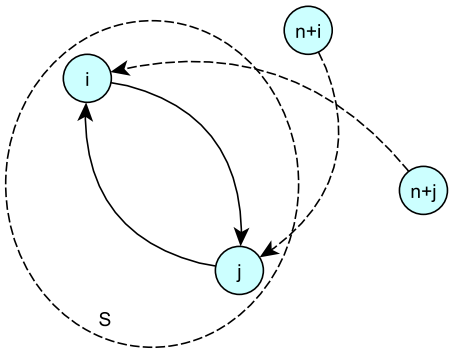

Figura 2: Exemplo da inequação (3.17). Fonte: Adaptado de Cordeau (2006).

de entrega e não possuem sucessores de entrega. Ainda, na segunda parcela da inequação não consideramos os arcos $x_{n+i,i}$ e $x_{n+j,j}$, pois estes podem ser previamente eliminados no pré-processamento. Agora, temos a seguinte situação:

- A primeira parcela da inequação equivale a $x(S) = x_{n+i,n+j} + x_{n+j,n+i}$;
- A segunda parcela é igual a $\sum_{i' \in S} \sum_{j' \in N \setminus S \cap \pi(S)} x_{i'j'}$ e temos que $i' = \{n+i, n+j\}$ e $j' = \{i, j\}$, ou seja, corresponde a $x_{n+i,j} + x_{n+j,i}$;
- A terceira parcela da inequação $\sum_{i' \in S \cap \pi(S)} \sum_{j' \in N \setminus S \setminus \pi(S)} x_{i'j'}$ e não há arcos nesta terceira parcela;
- O lado direito da inequação $|S| - 1$, temos que $|S| = 2$;
- A inequação resultante é dada por $x_{n+j,n+i} + x_{n+i,n+j} + x_{n+i,j} + x_{n+j,i} \leq 2 - 1 = 1$.

A Figura 3 ilustra os arcos deste exemplo. Os arcos pontilhados são os que são melhorados nesta inequação, ou seja, eles não pertencem à inequação clássica de eliminação de sub-rota (3.16). Além disso, a figura nos diz que, para satisfazer a inequação, devemos ter apenas um dos quatro arcos desenhados na solução ótima.

No caso de uma formulação em que o grafo seja direcionado, a restrição de eliminação de sub-rota clássica (3.16) pode ser apertada se levarmos em conta a orientação dos arcos.

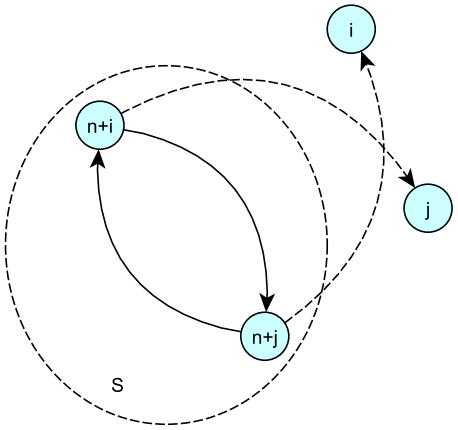

Figura 3: Exemplo da inequação (3.18). Fonte: Adaptado de Cordeau (2006).

Para um conjunto ordenado de nós $S = \{i_1,i_2,...,i_k\} \subseteq N$ com $k \geq 3$, Cordeau (2006) provou que as seguintes inequações propostas originalmente para o TSP (GRÖTSCHEL; PADBERG, 1985), também são válidas para o problema de coleta e entrega com janelas de tempo:

$$\sum_{j=1}^{k-1} x_{i_j,i_{j+1}} + x_{i_k,i_1} + 2\sum_{j=3}^{k} x_{i_1,i_j} + \sum_{j=4}^{k}\sum_{l=3}^{j-1} x_{i_j,i_l} + \sum_{h\in N\setminus S\cap\pi(S)} x_{i_1,h} \leq k-1, \tag{3.19}$$

$$\sum_{j=1}^{k-1} x_{i_j,i_{j+1}} + x_{i_k,i_1} + 2\sum_{j=2}^{k-1} x_{i_j,i_1} + \sum_{j=3}^{k-1}\sum_{l=2}^{j-1} x_{i_j,i_l} + \sum_{h\in N\setminus S\cap\sigma(S)} x_{h,i_1} \leq k-1. \tag{3.20}$$

Considere novamente o exemplo proposto por Cordeau (2006) para ilustrar a inequação (3.19). Seja o conjunto de três nós $S = \{i,j,k\} \subseteq P$. Uma possível restrição de eliminação de sub-rota obtida por $i_1 = i$, $i_2 = j$ e $i_3 = k$ é $x_{ij}+x_{jk}+x_{ki}+2x_{ji}+x_{n+j,i}+x_{n+k,i} \leq 2$. Esta situação está ilustrada na Figura 4, ou seja, dos seis arcos desenhados, podemos ter na solução ótima, apenas dois. Nesta situação, novamente os arcos pontilhados correspondem aos arcos que são melhorados em relação a eliminação de sub-rota clássica, enquanto que os arcos não pontilhados correspondem aos arcos da inequação clássica (3.16).

Restrições de Ordem Generalizadas:

As restrições de ordem generalizadas (*generalized order constraints*) foram primeira-

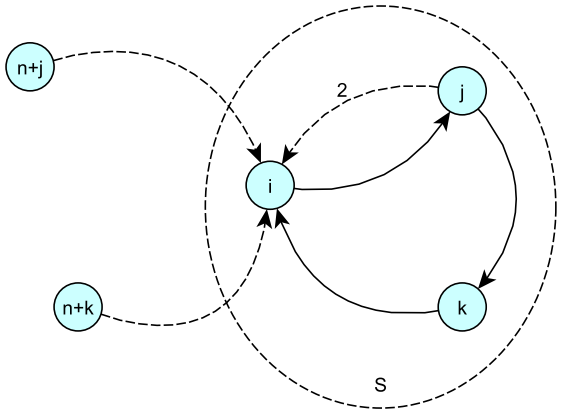

Figura 4: Exemplo da inequação (3.19). Fonte: Adaptado de Cordeau (2006).

mente propostas por Ruland e Rodin (1997) para o problema de coleta e entrega. Cordeau (2006) provou que estas inequações eram válidas também para o problema de coleta e entrega e janelas de tempo. Estas inequações tem o intuito de garantir que a coleta seja feita antes da entrega, isto é, de melhorar a restrição de precedência levando em conta pares de coleta e entrega, desde que esta coleta e esta entrega não sejam de uma mesma demanda. Antes, apresentaremos algumas definições. Sejam U_1,...,$U_s \subset N$ subconjuntos mutuamente disjuntos e sejam i_1,...,$i_s \in P$ as requisições, de modo que $0, 2n+1 \notin U_l$ (depósitos inicial e final, respectivamente) e i_l,$n+i_{l+1} \in U_l$ para $l = 1$,...,s, em que $i_{s+1} = i_1$. Sendo assim, temos que a seguinte inequação é válida para o PDPTW:

$$\sum_{l=1}^{s} x\left(U_l\right) \leq \sum_{l=1}^{s} |U_l| - s - 1. \tag{3.21}$$

A Figura 5 exemplifica como os conjuntos U estão definidos. Para $m = 3$, temos que $U_1 = \{i,n+j\}$, $U_2 = \{j,n+k\}$ e $U_3 = \{k,n+i\}$, e $i_1 = i$, $i_2 = j$ e $i_3 = k$. O lado esquerdo da inequação (3.21) para o exemplo representado na Figura 5 é igual a $\sum_{l=1}^{m} x\left(U_l\right) = x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k}$ e o lado direito igual a $(2+2+2)-3-1 = 2$ resultando em $x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k} \leq 2$. Assim, a soma dos fluxos em todos os arcos que estão dentro dos conjuntos U deve ser menor ou igual a 2. Isso implica que os arcos de no máximo dois conjuntos U, podem estar na solução ótima. De fato, uma solução com todos os arcos não pode ser factível (na rota de um único veículo ou em várias), pois a precedência seria violada. Sendo assim, da maneira como os conjuntos U estão definidos se permutarmos estes conjuntos, em algum momento teremos a violação da precedência.

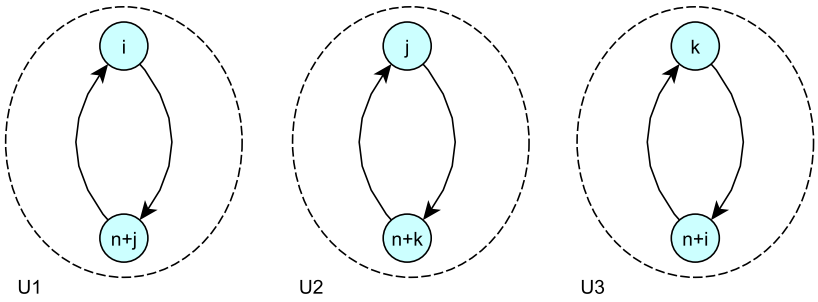

Figura 5: Exemplo dos conjuntos U. Fonte: Adaptado de Cordeau (2006).

Um segundo grupo de inequações chamado de precedência com quebra de ciclo (*precedence cycle breaking*) são propostas primeiramente por Balas et al. (1995) para o TSP assimétrico e nada mais são que uma melhoria nas restrições de ordem generalizadas. Cordeau (2006) provou que elas eram válidas também para o problema de coleta e entrega com janelas de tempo. Estas desigualdades são dadas por:

$$\sum_{l=1}^{s} x\left(U_l\right) + \sum_{l=2}^{s-1} x_{i_1,i_l} + \sum_{l=3}^{s} x_{i_1,n+i_l} \leq \sum_{l=1}^{s} |U_l| - s - 1, \tag{3.22}$$

$$\sum_{l=1}^{s} x\left(U_l\right) + \sum_{l=2}^{s-2} x_{n+i_1,i_l} + \sum_{l=2}^{s-1} x_{n+i_1,n+i_l} \leq \sum_{l=1}^{s} |U_l| - s - 1. \tag{3.23}$$

Observação: se enumerarmos os conjuntos U de maneiras diferentes, obtemos diferentes cortes.

Vamos mostrar a partir do exemplo representado pela Figura 5 como que os mesmos conjuntos U ficam em cada uma das duas inequações mais fortes. A Figura 6 refere-se à inequação (3.22). Temos os mesmos conjuntos U da inequação (3.21). Sendo assim, temos a seguinte situação:

- A primeira parcela da inequação é dada por $\sum_{l=1}^{m} x\left(U_l\right) = x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k}$;
- A segunda parcela é dada por $\sum_{l=2}^{m-1} x_{i_1,i_l} = x_{ij}$;
- A terceira parcela do lado esquerdo é $\sum_{l=3}^{m} x_{i_1,n+i_l} = x_{i,n+k}$;
- O lado direito continua igual a $(2+2+2)-3-1=2$;

- Portanto, a inequação (3.22) resulta em $x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k} + x_{ij} + x_{i,n+k} \leq 2$.

Na Figura 6, pode-se ver os arcos pontilhados que são os arcos acrescentados por esta restrição, ilustrando a diferença entre a restrição (3.21) e a restrição (3.22). Vemos através da figura que a precedência está sendo violada se tivermos mais de dois arcos na solução ótima, isto é, dos oito arcos desenhados, podemos ter apenas dois.

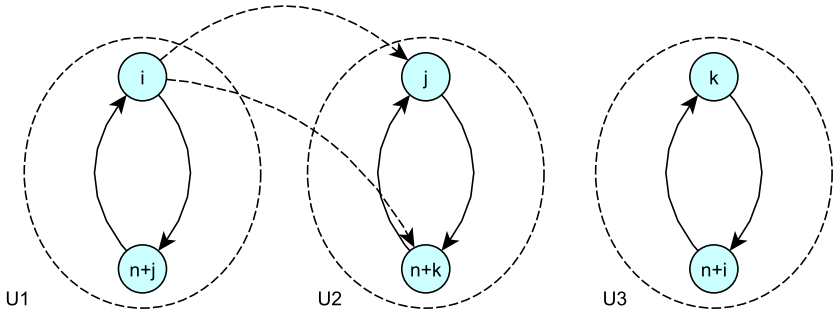

Figura 6: Exemplo da inequação (3.22). Fonte: Adaptado de Cordeau (2006).

O mesmo pode ser feito para a inequação (3.23). A Figura 7 exemplifica como fica esta restrição com a mesma definição dos conjuntos U, ou seja, quando $m = 3$. O arco pontilhado indica a diferença entre a inequação original (3.21) e esta, que é mais apertada. Então, temos o seguinte:

- A primeira parcela do lado esquerdo é igual a $\sum_{l=1}^{m} x(U_l) = x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k}$;
- A segunda parcela é igual a $\sum_{l=2}^{m-2} x_{n+i_1,i_l} = \varnothing$;
- A terceira parcela é igual a $\sum_{l=2}^{m-1} x_{n+i_1,n+i_l} = x_{n+i,n+j}$;
- O lado direito corresponde a $(2+2+2) - 3 - 1 = 2$;
- Sendo assim, temos que a inequação (3.23) equivale a $x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k} + x_{n+i,n+j} \leq 2$.

Neste caso, a precedência também pode ser violada, já que o nó $n+i$ pode ser visitado antes do nó i. Dos sete arcos desenhados, apenas dois podem estar na solução ótima. Cabe

fazer uma observação a respeito da inequação (3.23): o valor de m deve ser maior ou igual a 4 para que a segunda parcela do lado esquerdo tenha algum efeito, já que o somatório varia de $l = 2$ até $m - 2$, se $m < 4$ este somatório é vazio.

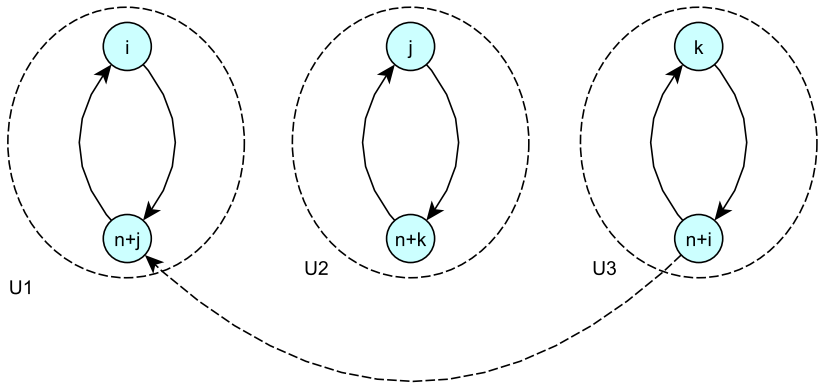

Figura 7: Exemplo da inequação (3.23). Fonte: Adaptado de Cordeau (2006).

Restrição de Capacidade:

A restrição dada a seguir pertence ao modelo PDPTW3 apresentado na Seção 2.2.2 (ROPKE et al., 2007). Para este modelo, esta restrição de capacidade deve garantir a factibilidade do problema, ou seja, sem ela não temos a garantia de que uma solução inteira respeite a capacidade do veículo. Já o modelo PDPTW2 possui duas restrições que garantem que a capacidade do veículo seja respeitada (2.20) e (2.22). Entretanto, na relaxação linear do problema, as restrições de capacidade descritas a seguir podem ajudar na melhoria do limitante fornecido. Sendo assim, podemos gerar cortes de capacidade através da restrição (2.31) do modelo PDPTW3, a fim de melhorar o limitante inferior fornecido pela relaxação linear do modelo PDPTW2. A inequação (2.31) do modelo PDPTW3 é dada por:

$$\sum_{i,j \in S} x_{ij} \leq |S| - \max\left\{1, \lceil |q(S)| / Cap \rceil\right\}, \forall S \subseteq N \setminus \{0, 2n+1\}, |S| \geq 2 \quad (2.31)$$

Temos que, para qualquer $S \subset P \cup D$, $q(S) = \sum_{i \in S} q_i$, ou seja, $q(S)$ corresponde ao somatório das demandas de todos os nós que estão em S. Sabemos que a melhor restrição de capacidade é dada se resolvermos o problema de empacotamento (*bin packing problem* - BPP). Em geral, é inviável computacionalmente resolvermos otimamente este problema, então consideramos um limitante inferior dado por $max\{1, \lceil |q(S)| / Cap \rceil\}$.

Observe que, caso não tivéssemos a parte $\lceil |q(S)| / Cap \rceil$ na restrição (2.31), isto re-

sultaria na eliminação de sub-rota clássica (3.16). Temos que $max\{1, \lceil |q(S)|/Cap \rceil\}$ nos dá a quantidade de veículos necessários para atender toda a demanda de S.

Restrição de Capacidade Mais Forte:

Este conjunto de inequações é utilizado também para melhorar o limitante inferior dos modelos de 2-índices apresentados anteriormente. Ropke et al. (2007) propõem esta nova classe de inequações que consideram os pares de nós $(k,n+k)$, tais que, a coleta do nó k é feita antes do veículo entrar no conjunto S, enquanto que a entrega $n+k$ é feita depois do veículo deixar este conjunto. Neste caso, o número de veículos que entram em S normalmente aumenta para acomodar toda a demanda de S. Formalizando, sejam $S,T \subset P \cup D$ dois subconjuntos disjuntos tais que $q(S) > 0$. Lembrando que $q(S) = \sum_{i \in S} q_i$ e define-se $x(S:T) = \sum_{i \in S} \sum_{j \in T} x_{ij}$. Definimos $U = \pi(T) \setminus (S \cup T)$. Sendo assim, temos a seguinte inequação:

$$x(S) + x(T) + x(S:T) \leq |S| + |T| - \left\lceil \frac{q(S) + q(U)}{Cap} \right\rceil. \tag{3.24}$$

Um exemplo é dado pela Figura 8 em que $S = \{i,j\}$, $T = \{n+k, n+l\}$ e $U = \pi(T) \setminus (S \cup T) = \{k,l\}$. Sendo assim, se $q_i = q_j = q_k = q_l = 1$ e a capacidade do veículo $Cap = 2$, então no máximo dois arcos da figura podem estar na solução ótima para que a capacidade do veículo seja respeitada. Os arcos da figura representam o lado esquerdo da inequação (3.24) e o lado direito corresponde a $|S| + |T| - \left\lceil \frac{q(S)+q(U)}{Cap} \right\rceil = 2 + 2 - 2 = 2$.

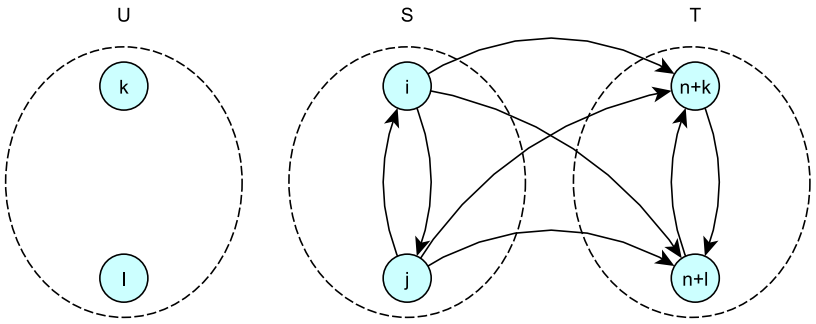

Figura 8: Exemplo da inequação (3.24). Fonte: Adaptado de Ropke et al. (2007).

Caminhos Infactíveis:

Estas inequações, como as inequações de capacidade (2.31), também pertencem ao modelo PDPTW3 e podem ser utilizadas para melhorar o limitante inferior do modelo PDPTW2. São utilizadas para verificar se algum caminho violou a restrição de janela de

tempo. Vamos relembrar algumas definições dadas na Seção 2.2.2: seja $\mathcal{R}$ o conjunto de todos os caminhos infactíveis em relação às janelas de tempo. Dado um $R \in \mathcal{R}$ seja $A(R)$ e $N(R)$ o conjunto de arcos e nós de R, respectivamente. Como R é um caminho e não uma rota, temos que $A(R)$ corresponde aos arcos que estão neste caminho. Sendo assim, temos que a inequação de caminhos infactíveis é dada por:

$$\sum_{(i,j)\in A(R)} x_{ij} \leq |A(R)| - 1 \quad \forall R \in \mathcal{R} \quad (2.32)$$

As restrições (2.32) podem ser tornar mais fortes introduzindo as chamadas restrições de torneio (*tournament constraints*) introduzidas primeiramente por Ascheuer et al. (2000). Se $R = (k_1,k_2,...,k_r)$ é um caminho infactível, então a próxima inequação vale:

$$\sum_{i=1}^{r-1} \sum_{j=i+1}^{r} x_{k_i,k_j} \leq |A(R)| - 1. \quad (3.25)$$

Em Ascheuer (2000) é mostrado que $A(R) - 1 = r - 2$, sendo que r é a quantidade de elementos em R. Sendo assim, podemos substituir o lado direito da inequação (2.32) por $r - 2$. As restrições (3.25) são mais apertadas que as restrições (2.32), pois possuem maior permutação dos nós que estão em R, ou seja, possui mais arcos no lado esquerdo da inequação. Por exemplo, suponha que $N(R) = \{1,5,3\}$ e que todos os arcos sejam iguais a 1, ou seja, $A(R) = 2$. Pela restrição (2.32) temos que

$$x_{15} + x_{53} \leq 2 - 1 = 1$$

Entretanto, a restrição (3.25) corresponde a

$$x_{15} + x_{13} + x_{53} \leq 2 - 1 = 1.$$

Ou seja, possui mais arcos do lado esquerdo e, com isto, se torna mais apertada. Além disso, essas inequações podem se tornar mais fortes se unirmos o par de coleta i com sua entrega $n + i$ no caminho R, ou seja, $R = (i,k_1,k_2,...,k_r,n + i)$. Se a desigualdade triangular é respeitada e R é infactível por não respeitar as janelas de tempo, então a seguinte inequação vale:

$$x_{i,k_1} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + x_{k_r,n+i} \leq |A(R)| - 2. \quad (3.26)$$

Neste caso, temos que r não é dado por todos os elementos de R e sim pelos elementos que estão entre os nós i e $n+i$. Desta maneira, o lado direito da inequação (3.26) pode ser substituído por $r-1$ (lembrando que este r difere da inequação (3.25)) (CORDEAU, 2006).

Restrições de Garfo:

As restrições do tipo garfo (*fork constraints*) servem para eliminar caminhos infactíveis. Por exemplo, se o caminho $R = (k_1,...,k_r)$ é factível, porém o caminho (i,R,j) é infactível para todo $i \in S$ e $j \in T$ com $S,T \subset N$, podemos associar a cada nó intermediário $k_2,..,k_{r-1}$ um conjunto de nós infactíveis e aplicar a restrição que Ropke et al. (2007) provaram ser válida para o PDPTW, que é dada a seguir. Seja R, como já definido, um caminho factível em G e $S,T_1,...,T_r \subset (P \cup D)$ subconjuntos tais que $k_j \notin T_{j-1}$ para $j = 2,...,r$. Se para algum inteiro $h \leq r$ considerarmos o par de nós $i \in S, j \in T_h$, o caminho $(i,k_1,...,k_h,j)$ é infactível, vale a seguinte inequação de saída do garfo (*outfork*):

$$\sum_{i \in S} x_{i,k_1} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + \sum_{h=1}^{r} \sum_{j \in T_h} x_{k_h,j} \leq r. \tag{3.27}$$

A Figura 9 (adaptada de Ropke et al. (2007)) indica que, quando $r = 3$ em cada conjunto, tenta-se encontrar um caminho que seja factível de k_1 para T_1, de k_2 para T_2 e assim por diante.

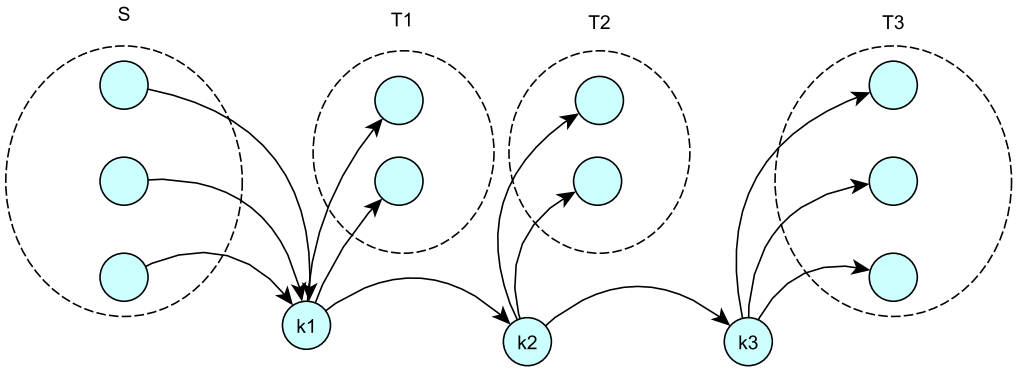

Figura 9: Exemplo da inequação (3.27). Fonte: Adaptado de Ropke et al. (2007).

Ropke et al. (2007) também provaram a validade das inequações de entrada do garfo (*infork*), que são similares à restrição anterior e são dadas a seguir:

$$\sum_{h=1}^{r} \sum_{i \in S_h} x_{i,k_h} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + \sum_{j \in T} x_{k_r,j} \leq r. \tag{3.28}$$

A Figura 10 exemplifica estas inequações em que se tenta encontrar um caminho factível de S_1 para k_1, de S_2 para k_2 e assim sucessivamente. É importante destacar que estas inequações podem ser utilizadas em qualquer contexto de roteamento de veículos em que o conceito de caminhos infactíveis esteja bem definido, em particular também para o problema de roteamento de veículos com janelas de tempo (ROPKE et al., 2007).

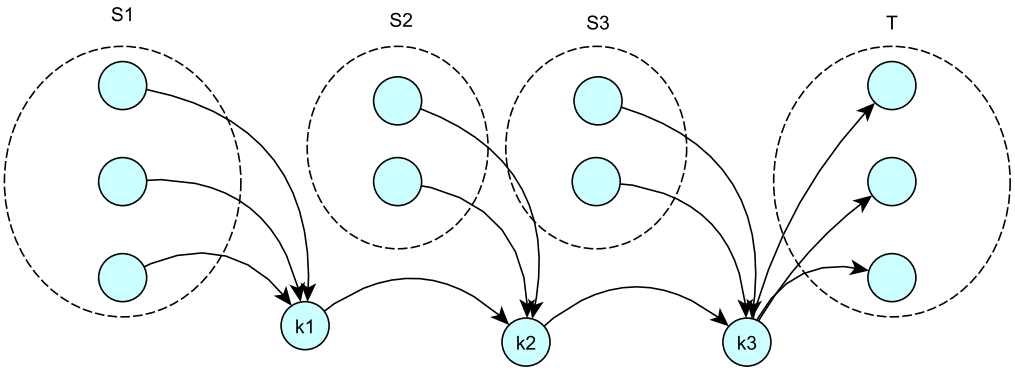

Figura 10: Exemplo da inequação (3.28). Fonte: Adaptado de Ropke et al. (2007).

Restrições de Alcance:

As restrições de alcance (*reachability constraints*) foram propostas inicialmente por Lysgaard (2006) para o VRPTW. Ropke et al. (2007) afirmaram que as mesmas são válidas para o PDPTW. Antes de introduzir estas inequações, vamos a algumas definições. Seja $\delta(S) = \delta^+(S) \cup \delta^-(S)$, em que $\delta^+(S) = \{(i,j) \in A | i \in S, j \notin S\}$ e $\delta^-(S) = \{(i,j) \in A | i \notin S, j \in S\}$, ou seja, estes conjuntos são definidos por arcos em que um elemento do arco está contido no conjunto S e o outro elemento está contido no conjunto complementar de S ($N \backslash S$). Ainda, para cada nó $i \in N$, seja $A_i^- \subset A$ o conjunto mínimo de arcos, tal que, qualquer caminho factível que inicie em 0 e termine em i, utiliza somente arcos de A_i^-. Seja A_i^+ o conjunto mínimo de arcos, tal que, qualquer caminho factível que se inicie em i e termine em $2n+1$, utiliza somente arcos de A_i^+. Considere o conjunto de nós T tais que, cada nó em T, deve ser visitado por um veículo diferente. Os autores comentam que este conjunto é dito ser conflitantes. Sendo assim, para qualquer conjunto de nós conflitantes T seja $A_T^- = \cup_{i \in T} A_i^-$ e $A_T^+ = \cup_{i \in T} A_i^+$. Para cada conjunto $S \subseteq P \cup D$ e cada conjunto do tipo conflitante $T \subseteq S$, as seguintes inequações valem:

$$x\left(\delta^-(S) \cap A_T^-\right) \geq |T|, \tag{3.29}$$

$$x\left(\delta^+(S) \cap A_T^+\right) \geq |T|. \tag{3.30}$$

Estas inequações forçam que os nós do conjunto T sejam visitados pelo menor cami-

nho. A interseção $\delta^{-}(S) \cap A_T^{-}$ contém os arcos que pertencem a algum caminho factível e, portanto, somente arcos em caminhos factíveis podem fazer parte do corte. No contexto do problema de coleta e entrega os nós podem ser ditos conflitantes não somente em relação às janelas de tempo, mas também em relação à questão da precedência dos nós. No caso do DARP, o tempo total de viagem também deve ser levado em conta na verificação se dois nós são ou não conflitantes.

Em Ropke et al. (2007) os autores compararam o desempenho de cada desigualdade válida dada anteriormente quando aplicadas separadamente na relaxação do problema PDPTW3 e PDPTW2 e também quando aplicadas todas juntas. Os resultados foram realizados para exemplares propostos em Cordeau (2006). As famílias de desigualdades válidas que tiveram maior impacto no limitante inferior foram do tipo garfo e de alcance. Alguns exemplares foram resolvidos otimamente em tempo computacional razoável por um método *branch-and-cut* com todas as famílias de desigualdades válidas.

No Capítulo 5, são apresentados os algoritmos de separação para algumas das restrições mostradas acima (caminhos infactíveis, restrição de capacidade, eliminação de sub-rota, restrições de ordem generalizadas e restrições de alcance). As demais restrições como do tipo garfo, capacidade mais forte, caminhos infactíveis mais forte e eliminação de sub-rota mais forte poderão ser acrescentadas ao método futuramente, a fim de melhorá-lo. Além disso, o Capitulo 5 mostra os algoritmos do tipo *branch-and-cut* implementados neste trabalho, para solução de problemas da empresa em estudo. A parte referente ao método *branch-and-price* é melhor discutida e apresentada no Capítulo 6.

4 *O problema de roteamento e programação de navios na indústria petrolífera*

Este capítulo primeiramente discute resumidamente os principais trabalhos sobre o problema de roteamento e programação de navios com ênfase nos problemas de coleta e entrega e métodos exatos. Em seguida, o problema de roteamento e programação de navios na indústria petrolífera é descrito, com base no estudo de caso realizado na empresa brasileira bem como suas características e diferenças em relação aos problemas clássicos presentes na literatura. Depois, a modelagem matemática é apresentada para o problema de coleta e entrega de óleos crus e janelas de tempo na indústria petrolífera. Por fim, são definidos os exemplares de testes reais fornecidos pela empresa brasileira e, em seguida alguns resultados computacionais referentes à validação do modelo matemático.

4.1 Roteamento e programação de navios

São muitos os trabalhos na literatura relacionados ao problema de roteamento e programação de veículos que envolvem modelagem matemática e métodos de solução. Porém, a literatura não é extensa para os trabalhos que envolvem especificamente roteamento e programação de navios. Segundo Christiansen et al. (2007), isto se deve a alguns fatores, como menor visibilidade (as pessoas estão acostumadas a ver caminhões, aviões e trens, porém não estão acostumadas a ver navios), o transporte marítimo é menos estruturado, há maior incerteza na tomada de decisões (devido às condições climáticas, por exemplo) e a indústria marítima é muito antiga, sendo mais difícil inserir novas ideias. A indústria que gira em torno de navios que trafegam nos oceanos possui o monopólio do transporte de grandes volumes entre continentes. Sendo assim, é importante que os custos envolvidos diminuam e que também, reduza-se o prejuízo causado na natureza, por exemplo, ao reduzir as operações de transporte (CHRISTIANSEN et al., 2004).

Segundo Hoff et al. (2010), na União Europeia o transporte de mercadorias em estradas aumentou 37% no período de 1995 a 2005, e o transporte de mercadorias no mar aumentou quase que a mesma proporção. Ainda segundo os mesmos autores, a alteração do volume de transporte acompanhou o aumento do PIB (produto interno bruto), devido ao crescimento econômico e também à crescente globalização. Outro fator que contribuiu com o aumento do transporte de mercadorias tanto nas estradas como nos oceanos é o fato de algumas empresas, que anteriormente negociavam preços e tarifas com outras empresas, hoje compartilharem informações a fim de aprimorar o desempenho global (ANDERSSON et al., 2010).

As atividades marítimas dependem crucialmente dos serviços que a frota de navios pode oferecer. Isto significa que há vezes em que um navio não pode atracar em determinado ponto devido à falta de estrutura local. Além disso, a frota de navios das empresas pode mudar com o passar do tempo, e a frota normalmente é heterogênea, diferenciando o tamanho dos navios, capacidade, velocidade, dentre outras características. Consequentemente, as empresas possuem problemas complexos e um planejamento extenso, variando entre níveis estratégicos, táticos e operacionais. Esta classe de problemas se diferencia do roteamento de veículos em geral, pois envolve outros tipos de características como, por exemplo, operações de atracação, problemas relacionados às marés, navios heterogêneos, tempo de viagem maior, o destino pode ser alterado durante o horizonte de planejamento, o navio não retorna ao depósito inicial, dentre outras (CHRISTIANSEN et al., 2007).

Existem três classes de problemas que envolvem roteamento e programação de navios no transporte marítimo: *liner*, *tramp* e industrial (CHRISTIANSEN et al., 2004). A primeira opera de acordo com um itinerário e horário pré-determinados e a demanda depende de suas programações como ocorre, por exemplo, em linhas de ônibus. Na classe *tramp*, os navios seguem os pontos em que as cargas estão disponíveis, de modo semelhante a um táxi. Nestes dois tipos de problemas, os operadores buscam maximizar os lucros por unidade de tempo. A última classe (industrial) normalmente possui cargas embarcadas e as embarcações utilizadas são diretamente controladas por uma empresa. Nesta classe, o objetivo é minimizar o custo de transportar as cargas. Diante destas características, o presente trabalho está inserido no modo industrial.

Um dos trabalhos pioneiros a abordar o problema de programação de navios foi em Dantzig e Fulkerson (1954), em que estudou-se o problema de programação de navios da marinha americana usados para o transporte de combustível. Os autores propuseram um modelo de programação linear e um método de solução exato para resolvê-lo com o

objetivo de minimizar o número de navios utilizados. Algum tempo depois, Appelgren (1969) estudou o problema de atribuição de cargas a navios com o objetivo de minimizar os custos relacionados às viagens. Este trabalho possui algumas semelhanças como, por exemplo, custos portuários, de canais e de carregamento e descarregamento e que envolvem o consumo do navio. Além disso, frota é heterogênea e há janelas de tempo para o carregamento e descarregamento (como no problema do estudo de caso). O modelo entretanto, é de programação linear e o autor utilizou o método de geração de colunas para resolvê-lo. Uma extensão de Appelgren (1969) foi o trabalho de Appelgren (1971), no qual o autor propõe métodos de programação inteira para resolver o problema, como o método de planos de corte o e o método *branch-and-bound*. O primeiro método não é capaz de resolver todos os casos teste. Sendo assim, o método *branch-and-bound* encontra solução inteira, mas não necessariamente a solução ótima (devido a forma como foi utilizado).

Em Mckay e Hartley (1974) foram apresentados dois modelos para resolver o problema de roteamento e programação de navios da marinha americana que transportam petróleo ao redor do mundo. Os modelos consideram multi-produto e janelas de tempo (semelhanças com o problema do estudo de caso). O método de solução é o seguinte: relaxa-se a integralidade do modelo e, após resolver o problema de programação linear resultante, a técnica de arredondamento seletivo é utilizada para a obtenção de uma solução inteira aproximada. Ronen (1983) e Ronen (1993) apresentaram revisões da literatura sobre o transporte marítimo, antes de 1983 e entre 1983 e 1993, respectivamente. Os trabalhos discutem as diferenças entre o roteamento e programação de veículos e navios. Além disso, analisam porque o roteamento de navios recebe menos atenção por parte dos pesquisadores. Então, é feita uma revisão sobre roteamento e programação e alguns modelos matemáticos são discutidos.

Brown et al. (1987) estudaram o problema de transporte de óleo cru para exportação (semelhança com o problema do estudo de caso) com o objetivo de encontrar as velocidades ótimas dos navios e otimizar roteiros e custos. Entretanto, a frota é homogênea e as datas de carregamento e entrega são pré-fixadas e o horizonte de planejamento é de até três meses (algumas diferenças). O problema é resolvido via geração de colunas, sendo que as colunas são programações viáveis para cada navio. O melhor resultado é para um exemplar com 50 cargas, 12 portos e 24 navios, o qual é resolvida em questão de minutos. Em Christiansen (1999), estudou-se o problema de roteamento e programação de navios e janelas de tempo em uma empresa que transporta amônia. O problema é encontrar rotas com custos mínimos de transporte de modo que a produção não pare e ainda manter os níveis de estoque das fábricas desejáveis. A frota é heterogênea a qual

possui custos, capacidades e características distintas (estas características representam várias semelhanças com o problema aqui tratado). O horizonte de planejamento é de 14 dias e pode se expandir em até 3 meses. O problema é formulado como um modelo de programação inteira mista e o método de decomposição *Dantzig-Wolfe* é utilizado para decompor o problema em subproblemas de rede independentes para cada navio. Os resultados mostraram que a abordagem proposta tem possibilidades de se tornar uma ferramenta para auxiliar a tomada de decisão da empresa em seu planejamento regional.

Sherali et al. (1999) estudaram a programação de navios para exportação de óleo cru e derivados do Kuwait para países na América do Norte, Europa e Japão. A frota é heterogênea, os navios podem carregar produtos diferentes e janelas de tempo são consideradas na coleta e entrega dos produtos. É proposto um modelo de programação inteira mista com rotas pré-geradas. Uma heurística baseada no modelo é proposta e utiliza horizonte de tempo rolante (diferença com o problema aqui tratado, não consideramos horizonte de tempo rolante). Os testes computacionais com até 20 navios (com 6 e 8 compartimentos), horizonte de tempo de 90 dias e 4 portos são bem sucedidos. Fagerholt (2000) apresentou um modelo para o problema de roteamento dos navios em que o objetivo está associado aos custos gerais do transporte e programação dos navios. Depois, Fagerholt (2001) considera um problema real de programação com uma frota heterogênea de navios para carga e descarga. No mesmo trabalho, são aplicados métodos heurísticos e técnicas de particionamento.

Christiansen et al. (2004) revisaram os problemas relacionados ao roteamento e programação de navios. O foco está na literatura publicada na década de 90. Esta revisão divide-se em algumas partes: problemas estratégicos, táticos operacionais, e por último, em aplicações navais. Este trabalho também mostra quais são as perspectivas futuras para o roteamento e programação de navios, bem como os benefícios da área. O problema de transporte marítimo de betume foi estudado em Persson e Göthe-Lundgren (2005). Os navios possuem compartimentos que permitem entregar produtos diferentes na mesma viagem. As rotas são previamente geradas e um modelo matemático é proposto, o qual é solucionado por um método exato que utiliza desigualdades válidas e geração de colunas.

Rocha et al. (2009) apresentaram um modelo matemático para o problema de alocação de petróleo da Petrobras, que envolve decisões relacionadas à frota de navios, tipos de petróleo transportado e em qual terminal o petróleo deve ser transportado. O objetivo é minimizar o custo total, que envolve custos da frota, estoque e penalidades por desviar a estratégia do planejamento, como também custos relacionados a utilizar frota de terceiros.

Este problema difere do problema do estudo de caso considerado nesta tese, pois envolve a tomada de decisões em um nível hierárquico superior ao aqui considerado, ou seja, as decisões no estudo de Rocha et al. (2009) servem como dado de entrada para o problema aqui estudado. Outro trabalho em que as decisões táticas foram tomadas em relação ao transporte de petróleo é Aizemberg et al. (2014). Os autores tomam decisões em relação ao tamanho do lote a ser transportado, o dia que isto acontecerá e também decisões em relação à rota de cada navio. Primeiramente os autores compararam a formulação proposta com outras já presentes na literatura e depois, propõem um método de geração de colunas baseado em uma heurística para o problema estudado. Em Kobayashi e Kubo (2010) estudou-se o problema de transporte marítimo de petróleo nos portos do Japão. O problema é modelado como problema de coleta e entrega, com multi-navios e janelas de tempo. O objetivo é minimizar o custo total de operação. Os autores propõem também uma heurística para o problema.

Hoff et al. (2010) fizeram uma revisão da literatura que descreve os aspectos industriais, características do roteamento de navios, classificação dos problemas e detalhes de estratégias encontradas na literatura para resolver os diversos problemas de roteamento e programação de navios. Outra revisão da literatura em transporte marítimo é o trabalho de Andersson et al. (2010), que enfatiza os processos e decisões relacionados à gestão de estoque e roteamento de navios e, principalmente, a combinação dessas atividades na perspectiva da pesquisa operacional. Em Hennig et al. (2012), estudou-se o problema de transporte de óleo cru entre pontos de oferta e demanda, com o objetivo de minimizar os custos de transporte. Outra característica importante é a questão do fracionamento da carga (*split*) ser permitido. Além disso, possui janelas de tempo, frota heterogênea, restrições de navios disponíveis e rotas, as quais são pré-geradas. Os autores propõem um modelo de programação linear inteira mista e os *softwares* de otimização de propósito geral se mostram eficazes na resolução deste modelo para exemplares de pequeno porte.

Outros trabalhos mais recentes que estudaram o roteamento e programação de navios, alguns em óleo cru, e resolvidos por métodos exatos são: Gribkovskaia et al. (2008), Hwang et al. (2008), Brønmo et al. (2010), Grønhaug et al. (2010), Stålhane et al. (2012) e Fagerholt e Ronen (2013).

Não há trabalhos em que todas as características do problema do estudo de caso foram estudadas (calado flexível, posicionamento dinâmico, impossibilidade de atracar, múltiplos depósitos e janelas de tempo, as quais são discutidas adiante). Alguns trabalhos estudaram problemas em que ocorreram apenas algumas destas características como, por

exemplo, em Christiansen (1999) em que tem-se o transporte marítimo de amônia, mas os estoques são mantidos por meio de restrições e não dentro das janelas de tempo; além de não possuir algumas restrições específicas como calado flexível, posicionamento dinâmico e impossibilidade de atracar. Em Hennig et al. (2012) tem-se uma descrição parecida com a do problema do estudo desta tese, porém este foi modelado de maneira diferente possibilitando o fracionamento da carga e, além disso, também não considera todas as características do problema do estudo de caso. Sendo assim, alguns trabalhos possuem algumas semelhanças e algumas diferenças sendo que nenhum contempla exatamente todas as características que são descritas a seguir.

4.2 Descrição do Problema

O problema de coleta e entrega e janelas de tempo na indústria petrolífera estudado nesta tese é inspirado em uma grande empresa que extrai óleo cru do mar no Brasil. Sua fundação data de Outubro de 1953 no governo de Getúlio Vargas e teve o objetivo de executar as atividades no setor petrolífero no Brasil. Sua instalação foi concluída em 1954 com a herança de duas refinarias do Conselho Nacional de Petróleo, uma na Bahia e outra em São Paulo. Nesta época, a produção era de 1,7% do consumo nacional. A primeira refinaria própria se deu em 1961 e a descoberta de petróleo no mar no Brasil se deu no ano de 1968, em Sergipe. Em 2013, a Petrobras produziu cerca de 2.598.300 barris por dia, um investimento de 84 bilhões de reais e um lucro líquido de 21 bilhões de reais. Suas principais atividades são: exploração e produção de petróleo e gás, petroquímica, refino de petróleo e gás, distribuição, geração de energia elétrica, produção de biocombustíveis, transporte e comercialização. O desafio é a produção do petróleo em águas cada vez mais profundas, isto é, *offshore*. Neste período, a Petrobras contou com 135 plataformas, sendo 80 fixas e 55 flutuantes, 15 refinarias e 237 navios, sendo 60 deles próprios (PETROBRAS, 2013).

Diante de números grandes, a atividade de roteirizar e programar os navios que saem das plataformas e carregam óleo cru até os terminais é uma tarefa difícil de ser realizada, porque além da escala do transporte, esta atividade possui muitas restrições e características próprias. Cabe ressaltar que este problema poderia ser modelado de outras maneiras como, por exemplo, como um problema de roteamento com controle de estoques (*inventory rounting problem* (AGRA et al., 2014)) ou como um problema de roteamento com fracionamento de cargas (*split pickup* - (HENNIG et al., 2012)). Neste trabalho optamos por modelar este problema como um problema de coleta e entrega, pois é a maneira como

os operadores da empresa enxergam a sistemática do transporte de óleo cru pelos navios.

O foco deste trabalho é na atividade de transporte de óleos crus das plataformas até os terminais. O problema de roteamento e programação de navios está inserido no contexto em que as origens e os destinos estão pré-fixados, ou seja, em um planejamento anterior, a empresa decide o quanto de óleo cru deve ser transportado de cada plataforma para cada terminal. Para a logística, o problema é decidir qual navio fará esta atividade e em qual momento. Além disso, cada navio começa e termina em seu "depósito próprio", que nada mais é que a posição (coordenadas latitude e longitude) que ocupa no exato momento de início e término de suas atividades dentro do horizonte de planejamento. Portanto, este é um típico problema de coleta e entrega com janelas de tempo e múltiplos depósitos. Cabe salientar que um estudo preliminar foi realizado com este mesmo problema em Rodrigues (2013), em que o autor desenvolveu um modelo matemático que serviu de ponto de partida para o modelo dessa tese.

Além disso, trata-se de um problema com multi-produto, ou seja, cada plataforma produz um óleo diferente e cada terminal demanda quantidades de óleos de cada plataforma especificamente. Por ser modelado como um problema de coleta e entrega, não precisa-se ter o índice referente ao produto, uma vez que cada terminal demanda uma quantidade específica de cada plataforma (óleo). Portanto, o multi-produto é considerado por meio dos modelos de coleta e entrega, com demandas para cada tipo de produto.

As rotas são designações de como uma ou mais demandas serão atendidas por um navio, ou seja, qual navio vai atender a qual demanda, tendo também que obedecer a um sequenciamento de escalas e cumprir determinadas operações. A Figura 11 representa um exemplo ilustrativo de uma rota de um navio na costa brasileira. Os nós em vermelho 1 e 6 representam os nós artificiais do navio, ou seja, depósitos inicial e final, respectivamente. Note que os nós 2 e 4 representam a mesma plataforma com latitude e longitude iguais, porém com janelas de tempo diferentes, trata-se de uma estratégia de modelagem que indica que é o mesmo ponto operacional, porém é representado como dois pontos (nós) diferentes no grafo. Cabe ressaltar que o termo ponto operacional não faz distinção de plataforma ou terminal. Sendo assim, a rota do navio inicia-se no nó artificial inicial 1, segue para a plataforma 2 que coleta o petróleo que será entregue logo em seguida no terminal 3. Depois, passa pela plataforma 4 e coleta o óleo cru que será entregue no terminal 5 e assim, o navio termina sua rota no nó artificial final 6. Embora não contemplado na ilustração desta figura, um navio pode visitar mais de uma plataforma em sequência antes de realizar a entrega em um ou mais terminais.

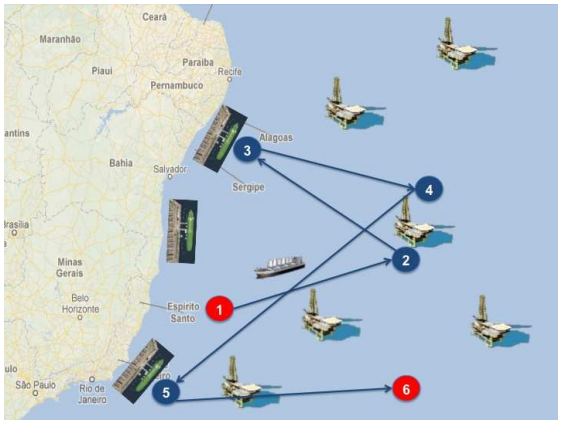

Figura 11: Exemplo de rota de um navio.

O produto tratado é o petróleo, que se subdivide em cerca de 50 tipos e assim tem-se aproximadamente 50 plataformas, uma vez que em geral cada plataforma produz um produto diferente. Os produtos podem ou não serem misturados num dado navio. Isto pode ocorrer, pois se o produto solicitado no terminal não tiver estoque disponível, então este pode ser substituído pela combinação de outros produtos que estejam disponíveis. Isto não é tratado na modelagem, uma vez que os operadores também não tem controle sobre os produtos que são misturados nos navios.

A frota de navios é heterogênea e de aproximadamente 120 navios para fins de transporte de petróleo e de outros produtos; destes, cerca de 50 são da própria empresa. Os navios que não pertencem à empresa são contratados ao longo de um ano, e este custo está inserido no custo fixo de cada navio. Cada navio possui capacidade e velocidade diferentes e outras características que serão abordadas neste capítulo. Além disso, não é em qualquer plataforma ou terminal que um navio pode atracar, devido a questões físicas de calado (que é a parte do navio que fica submersa na água) e LOA (*length overall* - comprimento do navio). A posição da frota, como já explicado, é identificada através das coordenadas latitude e longitude de cada navio. É necessário o cálculo de distâncias aos pontos operacionais e estimativas dos tempos de viagens em função das velocidades médias de cada navio. Como as velocidades médias dos navios não variam nos trajetos entre plataformas e terminais, é razoável considerar que todos os navios possuem a mesma velocidade média. A Figura 12 ilustra o calado e LOA de um navio.

São cerca de 50 plataformas *offshores* e estas se subdividem em outros 5 tipos: plataforma fixa, FPSO (*Floating, Production, Storage and Offloading*), navios-sonda que são

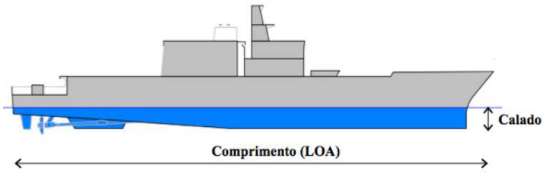

Figura 12: Características físicas do navio. Fonte: Rodrigues (2013)

projetados para a perfuração em alto mar, semissubmersível e, por último, plataforma autoelevável. A Figura 13 ilustra uma plataforma do tipo *offshore* semissubmersível.

Figura 13: Exemplo de uma plataforma *offshore*. Fonte: Culturamix (2013)

Em relação aos terminais, existem cerca de 8 com 20 berços no total, ou seja, cada terminal pode conter um ou mais berços, que o lugar onde de fato o navio atraca para descarregar o produto. Os berços podem ser diferentes em cada terminal, isto é, comprimento e largura distintos. Sendo assim, não é qualquer navio que pode atracar em qualquer berço, devido às restrições físicas. A Figura 14 mostra o terminal de São Sebastião em São Paulo; nela podemos ver dois navios atracados.

Um problema da empresa desse estudo de caso se encontra em manter os níveis de estoque das plataformas e terminais. As plataformas possuem um estoque a ser mantido, um mínimo e outro máximo, não podendo diminuir o estoque mínimo (por questões de segurança) e nem ultrapassar o estoque máximo (pelo alto custo de oportunidade associado - a plataforma não pode parar sua produção, pois representa um grande investimento da empresa). A empresa controla os níveis de estoque das plataformas e terminais por meio das janelas de tempo. Se as janelas forem respeitadas, então consequentemente os níveis de estoque são mantidos dentro dos níveis aceitáveis. Como não se tem restrições específicas de controle de estoque, isto é garantido através do cumprimento das janelas

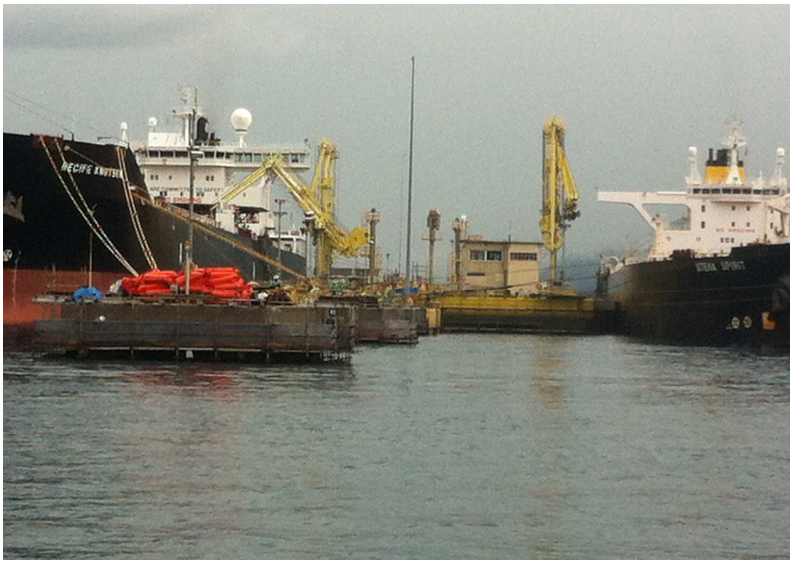

Figura 14: Terminal de São Sebastião

de tempo ao longo do horizonte de planejamento. Este horizonte de planejamento comumente é de poucas semanas de operação, o que corresponde a aproximadamente algumas dezenas de solicitações, ou seja, dezenas de pares de coleta e entrega. Com todas estas características e com janelas de tempo apertadas, o roteamento e programação de navios não é uma tarefa fácil para os programadores da empresa.

Os tamanhos dos navios variam muito, sendo os menores com capacidade em torno de 50000 m^3 e os maiores em torno de 170000 m^3. Além disso, nem todos os navios estão disponíveis no início do horizonte de tempo, pois podem estar em manutenção ou carregando algum produto. Sendo assim, algumas janelas de tempo dos navios se iniciam depois do horizonte se iniciar, ou seja, em períodos posteriores ao seu início. As demandas também variam, em algumas plataformas a produção é baixa, chegando a se ter uma demanda por óleo de 10000 m^3, enquanto outras plataformas possuem taxa de bombeamento mais elevada, chegando a 100000 m^3 de demanda por óleo. Além disso, uma mesma plataforma pode ter várias demandas num mesmo horizonte de planejamento de 2 semanas aproximadamente. Por exemplo, uma plataforma pode chegar a ter 8 demandas de produtos em torno de 2 semanas de produção. As regiões Norte e Nordeste não entram no planejamento por serem muito específicas, pois possuem navios dedicados e plataformas que possuem demandas muito bem definidas.

Portanto, trata-se de um problema de coleta e entrega e janelas de tempo, múltiplos depósitos e frota heterogênea, além de outras restrições específicas do problema. Este problema possui as restrições clássicas do PDPTW já discutidas neste texto como, por

exemplo, se um navio entra em um ponto operacional, este mesmo navio deve sair deste ponto, que a coleta e a entrega de uma demanda específica deve ser atendida pelo mesmo navio, dentre outras. Além disso, possui restrições específicas para o estudo caso descritas a seguir:

- Impossibilidade de atracação: alguns navios não podem atracar em certos pontos operacionais, sejam plataformas ou terminais, devido às restrições físicas como calado e LOA. Sendo assim, temos uma matriz A_{ik} que indica se o navio k pode ou não atracar no ponto operacional i;

- Calado flexível: mesmo quando há a exigência de que um navio k não deva atracar em um determinado ponto operacional, digamos i, em alguns casos é possível que este navio k tenha permissão em i se estiver apenas parcialmente carregado, ou seja, somente com uma porcentagem de sua capacidade máxima. Isto é aqui denominado de restrição de calado flexível. Na linguagem dos operadores, isto significa que o calado do navio é flexibilizado para que este possa atracar em determinados pontos operacionais, mas nestes casos o navio não pode estar totalmente carregado;

- Posicionamento dinâmico: alguns navios e alguns navio-plataformas, que são navios adaptados para a exploração de petróleo, possuem um sistema operacional denominado posicionamento dinâmico (DP - *dynamic positioning*). Os navios adaptados possuem a denominação de unidade flutuantes de produção (FPS - *Floating Production Systems*). Apenas as plataformas deste tipo (navio-plataforma) possuem este sistema. O DP controla automaticamente a posição do navio através de um complexo sistema composto por muitas variáveis, o que resulta em um posicionamento mais preciso do mesmo. As plataformas e navios que possuem este sistema estão sujeitos a determinadas regras para que a atracação seja possível. Estas regras combinam, por exemplo, a possibilidade de um navio com posicionamento dinâmico atracar em uma plataforma convencional (sem posicionamento dinâmico), ou atracar em uma plataforma com posicionamento dinâmico. Estas regras implicam que, por vezes, a atracação somente é possível se o navio possuir uma porcentagem de carga a bordo. Estas regras são discutidas mais a frente neste texto;

- Cada navio deve iniciar sua rota em seu "depósito inicial" e terminar em seu "depósito final". Sendo assim, cada navio tem um nó de partida e um nó de chegada pré-definidos, que são nós artificiais que indicam a latitude e longitude de cada navio no instante de início e término do horizonte de planejamento. O "depósito final" é

caracterizado pelo lugar em que o navio termina o horizonte de tempo, então não há distância entre o último terminal visitado e o depósito final do navio;

- Penalidade por visitas consecutivas: sempre que um navio visitar uma plataforma e em seguida, visitar outra plataforma que seja diferente da primeira, isto é penalizado na função objetivo. O intuito é fazer com que o navio colete e entregue sem visitar duas plataformas diferentes consecutivamente, devido às questões de segurança. Como em algumas situações isto é inviável, ocorre apenas uma penalização na função objetivo e não a proibição em si. Um dado navio pode realizar duas coletas consecutivas em uma mesma plataforma, desde que as janelas de tempo destas duas demandas estejam sobrepostas, e isto não é contabilizado como visita consecutiva;
- Função objetivo: modela os custos relacionados ao consumo de combustível, atracações realizadas e também penaliza visitas consecutivas a duas plataformas diferentes. Note que esta função objetivo é diferente das apresentadas em problemas clássicos de coleta e entrega da literatura.

Mais detalhes e informações sobre esse problema do estudo de caso dessa tese também podem ser encontrados em Rodrigues (2013).

4.3 Modelagem Matemática

Esta seção tem o objetivo de descrever a modelagem matemática para o problema descrito anteriormente de coleta e entrega e janelas de tempo na indústria petrolífera. Este modelo é baseado no modelo PDPTW1 apresentado na seção de problemas de coleta e entrega do capítulo anterior (Seção 2.2). Uma estratégia já mencionada e que cabe ressaltar aqui é que, na modelagem, um mesmo ponto operacional pode aparecer várias vezes ao longo do horizonte de planejamento. Isto ocorre porque as plataformas continuam sua produção e podem assim, encher novamente após um alívio (descarga) durante o horizonte. Sendo assim, suponhamos que a plataforma i tenha que ser descarregada (também chamada "aliviada" pelos operadores) em dois momentos diferentes dentro do horizonte de planejamento. Então, duplicamos o nó i como se fosse um outro ponto operacional, digamos j, entretanto com a mesma latitude e longitude do nó i. Os conjuntos, parâmetros e variáveis são dados a seguir.

Conjuntos

- K é o conjunto dos navios;

- P é o conjunto de nós que representam as coletas nas plataformas (origens) ($i \in P$);
- D é o conjunto de nós que representam as entregas nos terminais (destinos). Os nós D são enumerados da seguinte forma: para cada $i \in P$, é criado o nó $(n+i)$ o qual o nó $i \in P$ refere-se à origem i e o nó $(n+i) \in D$ refere-se ao respectivo destino;
- $ST = \{s_1, s_2,...,s_n\}$ é o conjunto dos nós artificiais que indicam o depósito inicial de cada navio, sendo s_k (*start node*) o nó inicial do navio $k \in K$;
- $EN = \{en_1, en_2,...,en_n\}$ é o conjunto dos nós artificiais que indicam o depósito de chegada de cada navio, sendo en_k (*end node*) o nó final do navio $k \in K$;
- O conjunto $N = P \cup D \cup ST \cup EN$ representa todos os nós da rede;
- O conjunto $A\,\{(i,j) : i,j \in N\}$ representam todos os arcos da rede.

Parâmetros

- $n = |P| = |D|$ número total de coletas (ou entregas);
- t_{ij} é o tempo de deslocamento em milhas por nós (horas) do nó $i \in ST \cup P \cup D$ para o nó $j \in EN \cup P \cup D$ - independe do navio k, pois consideramos que todos os navios possuem a mesma velocidade média;
- d_i é o tempo de serviço em horas do nó i, $\forall i \in P \cup D$;
- e_i é o início da janela de tempo em horas referente ao nó $i \in N$;
- l_i é o fim da janela de tempo em horas referente ao nó $i \in N$;
- Cap_k é a capacidade do navio k em m^3, $\forall k \in K$;
- q_i é a demanda do nó $i \in P \cup D$ em m^3 (pode ser positiva ou negativa). Se for positiva, indica que o ponto operacional i produz q_i unidades do produto. Se for negativa, indica que o ponto operacional i consome q_i unidades do produto. Por convenção $q_{n+i} = -q_i$;
- cm_k é o consumo de combustível do navio k em movimento, $\forall k \in K$;
- cs_k é o consumo de combustível do navio k enquanto está parado (*stand-by*), $\forall k \in K$;
- ca_j é o custo por atracação no nó j em reais;

- v é a velocidade média do navio em nós;
- $dist_{ij}$ é a distância em milhas náuticas entre o nó $i \in (ST \cup P \cup D)$ e o nó $j \in (EN \cup P \cup D)$. Se $dist_{ij} = 0$, então isto significa que os nós i e j representam o mesmo ponto operacional (mesma plataforma ou mesmo terminal);
- $A_{ik}, i \in P \cup D, k \in K$, é igual a 1 se o navio k não pode atracar no ponto operacional i, e 0 caso contrário;
- C_{DP} é um vetor que indica as plataformas que possuem posicionamento dinâmico. Um elemento de C_{DP} é igual a 0 se é permitido a atracação de um navio convencional ou um navio que tenha posicionamento dinâmico; e é igual a 1 se é permitido atracar somente navios com DP. Portanto, se uma plataforma não possui posicionamento dinâmico, então ela aceita somente navios que tenham DP;
- K_{DP} é um vetor que indica os navios k que possuem posicionamento dinâmico. Um elemento de K_{DP} é igual a 1 se o navio possui posicionamento dinâmico, e igual a 0, caso contrário;
- CF_{ik} é a matriz de calado flexível $\forall i \in N, k \in K$. É positiva se o navio k pode atracar em i, mediante ao navio conter no máximo a bordo $CF_{ik}\%$ de sua capacidade máxima;
- α_1 é a porcentagem de carga máxima permitida para o navio com DP atracar em qualquer plataforma $j \notin C_{DP}$;
- α_2 é a porcentagem de carga máxima permitida para o navio convencional atracar em uma plataforma $j \notin C_{DP}$;
- β é a penalização que ocorre na função objetivo por visitas consecutivas a duas plataformas diferentes;
- M é um número suficientemente grande.

Variáveis

- x_{ijk} é uma variável binária igual a 1 se e somente se o navio k percorre o arco (i,j) com $i,j \in N$;
- B_{ik} indica o instante de início de serviço no nó $i \in N$ pelo navio $k \in K$, se o k visitou i e 0 caso contrário;

- Q_{ik} é a quantidade de carga no navio k no instante imediatamente após sua visita ao nó i, $\forall i \in N, k \in K$, se k visitou i, e 0 caso contrário. Assume-se que o navio começa e termina sua rota vazio;

- VC_{ijk} é uma variável inteira que assume valor positivo se o navio k visita a plataforma i e, em seguida, a plataforma j com $dist_{ij} > 0$, ou seja, se i e j representam plataformas diferentes no grafo.

O modelo completo para representar o problema de coleta e entrega de óleos crus das plataformas para os terminais com janelas de tempo e frota heterogênea é dado por:

$$\text{Min} \sum_{i\in N}\sum_{j\in N}\sum_{k\in K}(cm_k - cs_k)\, x_{ijk}\frac{dist_{ij}}{v} + \sum_{i\in N}\sum_{\substack{j\in P\cup D \\ dist_{ij}>0}}\sum_{k\in K} ca_i x_{ijk} + \sum_{i\in D}\sum_{j\in EN}\sum_{k\in K} ca_i x_{ijk} + \sum_{i\in P}\sum_{j\in P}\sum_{k\in K}\beta * VC_{ijk} \tag{4.1}$$

s.a

$$\sum_{j\in(P\cup D\cup EN)}\sum_{k\in K} x_{ijk} = 1 \qquad \forall i \in (ST \cup P \cup D) \tag{4.2}$$

$$\sum_{i\in(P\cup D\cup ST)}\sum_{k\in K} x_{ijk} = 1 \qquad \forall j \in (P \cup D \cup EN) \tag{4.3}$$

$$\sum_{j\in(P\cup\{en_k\})} x_{s_k jk} = 1 \qquad \forall k \in K \tag{4.4}$$

$$\sum_{i\in(\{s_k\}\cup D)} x_{ien_k k} = 1 \qquad \forall k \in K \tag{4.5}$$

$$\sum_{i\in N}\sum_{k\in K} x_{ijk} = 0 \qquad \forall j \in ST \tag{4.6}$$

$$\sum_{j\in N}\sum_{k\in K} x_{ijk} = 0 \qquad \forall i \in EN \tag{4.7}$$

$$\sum_{i\in(P\cup D\cup\{s_k\})} x_{ihk} - \sum_{(j\in P\cup D\cup\{e_k\})} x_{hjk} = 0 \qquad \forall h \in P \cup D; k \in K \tag{4.8}$$

$$e_i\left(\sum_{j\in N} x_{jik}\right) \leq B_{ik} \leq l_i\left(\sum_{j\in N} x_{jik}\right) \qquad \forall i \in (P \cup D \cup EN)\,; k \in K \tag{4.9}$$

$$e_i\left(\sum_{j\in N} x_{ijk}\right) \leq B_{ik} \leq l_i\left(\sum_{j\in N} x_{ijk}\right) \qquad \forall i \in ST; k \in K \tag{4.10}$$

$$x_{ijk}\left(B_{ik}+t_{ij}+d_i-B_{jk}\right)\leq 0 \qquad \forall i\in(ST\cup P\cup D);\ j\in(P\cup D\cup EN);\ k\in K \qquad (4.11)$$

$$B_{n+h,k}\geq B_{h,k} \qquad \forall h\in P; k\in K \qquad (4.12)$$

$$Q_{jk}\geq\left(Q_{ik}+q_j\right)x_{ijk} \qquad \forall i\in(ST\cup P\cup D);\ j\in(P\cup D\cup EN);\ k\in K \qquad (4.13)$$

$$x_{ijk}=0 \qquad \forall i,j\in N;\ k\in K: A_{ik}=1;\ CF_{ik}\leq 0 \qquad (4.14)$$

$$\sum_{i\in(ST\cup P\cup D)} x_{ihk}=\sum_{j\in N} x_{j,n+h,k} \qquad \forall h\in P; k\in K \qquad (4.15)$$

$$Q_{jk}\leq Cap_k\sum_{i\in(\{s_k\}\cup P\cup D)} x_{ijk} \qquad \forall j\in(P\cup D\cup\{en_k\});\ k\in K \qquad (4.16)$$

$$Q_{s_k,k}+Q_{en_k,k}=0 \qquad \forall k\in K \qquad (4.17)$$

$$Q_{jk}\leq\left(CF_{jk}Cap_k+q_j\right)+\left(1-\sum_{i\in(P\cup D\cup\{s_k\})} x_{ijk}\right)M \qquad \forall j\in D; k\in K: A_{jk}=1 \qquad (4.18)$$

$$Q_{jk}\leq\left(\alpha_1 Cap_k+q_j\right)+\left(1-\alpha_1\right)Cap_k\left(1-\sum_{\substack{i\in(P\cup D\cup\{s_k\})\\ dist_{ij}>0}} x_{ijk}\right) \qquad \forall j\in P; K_{DP_k}=1 \qquad (4.19)$$

$$Q_{jk}\leq\left(\alpha_2 Cap_k+q_j\right)+\left(1-\alpha_2\right)Cap_k\left(1-\sum_{\substack{i\in(P\cup D\cup\{s_k\})\\ dist_{ij}>0}} x_{ijk}\right) \qquad \forall C_{DP_j}=0; K_{DP_k}=0 \qquad (4.20)$$

$$\sum_{i\in N} x_{ijk}=0 \qquad \forall C_{DP_j}=1; K_{DP_k}=0 \qquad (4.21)$$

$$x_{ijk}\leq VC_{ijk} \qquad \forall i,j\in P: dist_{ij}>0;\ k\in K \qquad (4.22)$$

$$VC_{ijk}\geq 0 \qquad \forall i,j\in N; k\in K \qquad (4.23)$$

$$Q_{ik} \geq 0 \qquad \forall i \in N; k \in K \tag{4.24}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in (ST \cup P \cup D); \; j \in (P \cup D \cup EN); \; k \in K \tag{4.25}$$

$$B_{ik} \geq 0 \qquad \forall i \in N; k \in K \tag{4.26}$$

A função objetivo (4.1) reflete os critérios a serem otimizados de acordo com os operadores da empresa. É composta pelos custos associados ao consumo de combustíveis dos navios, a quantidade de atracações realizadas e também penaliza duas visitas consecutivas a plataformas diferentes. A primeira parcela corresponde ao custo em relação ao tempo que o navio se movimenta, isto é, o custo variável. Este custo variável é dado pela diferença entre o custo do navio em movimento e o custo do navio parado definidos pelo contrato de afretamento dos navios. O custo do navio parado é fixo, ou seja, o navio tem um contrato e este indica o custo do navio por ano, independente da sua utilização. Por exemplo, se um navio possui custo de $ 42 unidades em movimento e custo de $ 32 parado significa que o valor a ser pago é de $ 10 unidades dadas em proporção do tempo, pois os $ 32 relacionados ao custo parado já foram contabilizados como custo fixo, isto é, já foram pagos. As segunda e terceira parcelas indicam o custo relacionado às atracações nos pontos operacionais. Há duas parcelas indicando o mesmo custo devido à necessidade de diferenciar quando um navio já se encontra próximo do ponto operacional, ou seja, quando $dist_{i,j} = 0$ e quando esta distância é maior que zero. Se um mesmo navio atraca em um ponto operacional e realiza duas coletas (ou entregas) num mesmo ponto, então este custo é contabilizado apenas uma vez. Isto pode ocorrer se as demandas de uma mesma plataforma ou terminal possuírem janelas de tempo que estejam sobrepostas. A última parcela, refere-se à uma penalização de β unidades caso o navio opte por visitar duas plataformas diferentes consecutivamente. Por vezes, uma mesma plataforma ou terminal pode ter diferentes demandas dentro do horizonte de tempo, e algumas com janelas de tempo sobrepostas. No caso em que uma mesma plataforma possui duas demandas com janelas de tempo que se sobrepõem, a penalização nestas duas visitas consecutivas à mesma plataforma não é contabilizada. Cabe salientar que existe um custo fixo relacionado aos navios (definido por contrato), mas como este não é otimizado, ele não faz parte da função objetivo.

As restrições (4.2) asseguram que existe exatamente um arco que sai de i e as restrições (4.3) garantem que existe exatamente um arco que entra em j; as duas juntas garantem

que todos os nós sejam visitados. Em (4.4) e (4.5) é assegurado que todos os navios saiam de seus depósitos iniciais e retornem aos seus depósitos finais, respectivamente. Todo navio parte de seu nó inicial e não pode mais retornar a ele, conforme imposto pela restrição (4.6). De forma análoga, um navio não pode partir de seu depósito final, o que é garantido pela restrição (4.7). A restrição (4.8) garante a conservação de fluxo dos navios, ou seja, que se algum navio chegou em um nó i, este mesmo navio tem que sair do nó i.

As janelas de tempo são asseguradas pelas restrições (4.9) e (4.10). Além disso, a restrição (4.10) permite que alguns navios possam iniciar sua rota após o início do horizonte de planejamento. Em (4.11) é imposto que o instante de início de serviço no nó j tem que ser maior ou igual ao instante de início de serviço no nó i, mais o tempo de serviço no nó i e o tempo de viagem entre os dois nós, se o navio k viaja de i para j. O navio k deve coletar primeiro antes de entregar a devida demanda, o que é garantido em (4.12). As restrições (4.13) garantem o atendimento da demanda pelo navio e em (4.14), que o navio não atraca em pontos operacionais em que existe alguma restrição física. Em (4.15) assegura-se que se o navio k visita o nó h, então ele precisa entregar a carga no nó $n+h$. A capacidade máxima do navio é imposta em (4.16). Em (4.17) assegura-se que o navio começa e termina vazio.

Em (4.18) garante-se que se o navio não for autorizado a atracar em um determinado ponto operacional, porém seu calado for flexibilizado, então permite-se atracá-lo neste ponto caso o navio possua até uma determinada porcentagem de carga a bordo. Em (4.19)-(4.21) garante-se que o posicionamento dinâmico seja respeitado. Na restrição (4.22) contabiliza-se duas visitas consecutivas a plataformas diferentes. Por fim, as restrições (4.23)-(4.26) garantem o domínio das variáveis de decisão.

Esta formulação é não-linear pelas restrições (4.11) e (4.13). A primeira é linearizada pela seguinte restrição:

$$B_{jk} \geq B_{ik}+d_i+t_{ij}+(x_{ijk}-1)\,M,\ \forall i \in (ST \cup P \cup D)\,, j \in (EN \cup P \cup D)\,, k \in K, \quad (4.27)$$

E a restrição (4.13) é linearizada pela restrição:

$$Q_{jk} \geq Q_{ik} + q_j + (x_{ijk}-1)\,M,\ \ \forall i \in (ST \cup P \cup D)\,, j \in (EN \cup P \cup D)\,, k \in K. \quad (4.28)$$

As regras de atracação utilizadas pela empresa para o posicionamento dinâmico são as seguintes:

- Se a plataforma é uma plataforma com posicionamento dinâmico:

  - Se é um navio que possui DP, então este navio pode possuir a bordo uma carga com até 50% de sua capacidade para atracar;
  - Se o navio for convencional, então pode possuir a bordo uma carga com até 30% de sua capacidade para atracar.

- Se a plataforma for convencional:

  - Se o navio possui DP, então o navio pode possuir uma carga a bordo de até 50% de sua capacidade para atracar;
  - Se o navio for convencional, então consideramos que este não pode atracar nesta plataforma.

O modelo proposto contempla de forma integrada todas as características descritas no início desta seção, que são específicas do estudo de caso aqui abordado. O modelo resultante corresponde a um problema de programação inteira mista, podendo ser resolvido pelos *softwares* de otimização disponíveis no mercado como, por exemplo, o IBM CPLEX e o GUROBI. Embora tais *softwares* tenham evoluído bastante nos últimos anos, o modelo proposto é bastante desafiador, conforme discutido na sequência.

4.4 Pré-processamento

Antes de resolver o modelo matemático proposto, podemos eliminar diversos arcos que sabemos a *priori* que não poderão existir em uma solução factível como, por exemplo, um navio não pode sair de i com destino ao próprio nó i. Esta pré-fixação de variáveis específica é feita impondo-se x_{ijk}, em que $x_{iik} = 0$. Entretanto, existem outras fixações com estas variáveis e também com outras variáveis do modelo. Por simplicidade considera-se fixação de variáveis e não a eliminação destes arcos no grafo. Algumas das fixações são ilustradas a seguir para facilitar seu entendimento. Para tanto, considere um grafo formado por apenas dois navios, então os nós 1 e 2 representam os depósitos artificiais iniciais dos navios 1 e 2, respectivamente, os nós 3 e 4 representam as coletas, os nós 5 e 6 as entregas e, por fim, os nós 7 e 8, os depósitos artificiais finais dos navios 1 e 2, respectivamente. A Figura 15 ilustra como este grafo é formado.

As regras de fixação de variáveis são dadas por:

1. Essa seguinte fixação evita que algum nó não tenha sido corretamente alocado ao

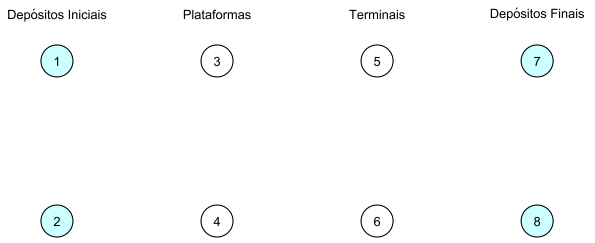

Figura 15: Representação do grafo para as fixações de variáveis.

conjunto ST. A Figura 16 mostra que esta fixação evita que o navio 1 saia do depósito inicial 2.

$x_{ijk} = 0, \ \forall i \in ST, j \in N, k \in K$ e $i \neq s_k$;

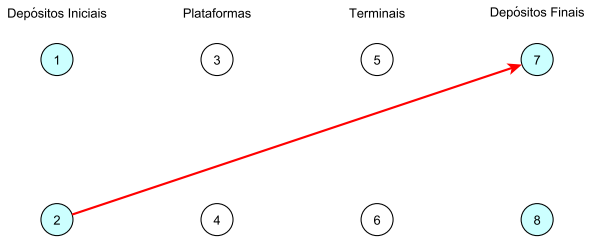

Figura 16: Exemplo da fixação de variável.

2. Essa próxima fixação evita que algum nó não tenha sido corretamente alocado ao conjunto EN:

   $x_{ijk} = 0, \ \forall j \in EN, i \in N, k \in K$ e $j \neq en_k$;

3. O navio não pode passar antes pela entrega e depois pela respectiva coleta. A Figura 17 ilustra que o navio não pode percorrer o arco (5,3) já que 5 é a respectiva entrega de 3.

   $x_{n+h,h,k} = 0, \ \forall h \in P, k \in K$;

4. Não existe arco entre o nó de saída de um navio para uma entrega, considerando que o navio comece vazio:

   $x_{ijk} = 0, \ \forall i \in N, j \in D, k \in K$ e $i = s_k$;

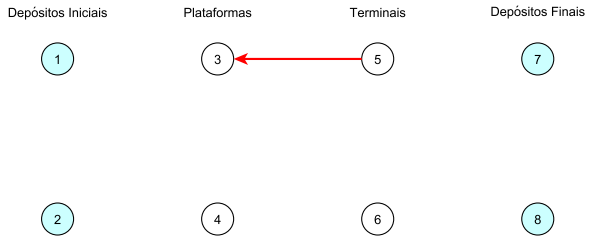

Figura 17: Exemplo da fixação de variável.

5. Não existe arco entre uma coleta e um nó de chegada de um navio. Isto é representado pela Figura 18 que ilustra que não podemos ter o arco (3,7) já que 3 é uma coleta e 7 o depósito final do navio 1.

   $x_{ijk} = 0, \ \forall i \in P, j \in N, k \in K$ e $j = en_k$;

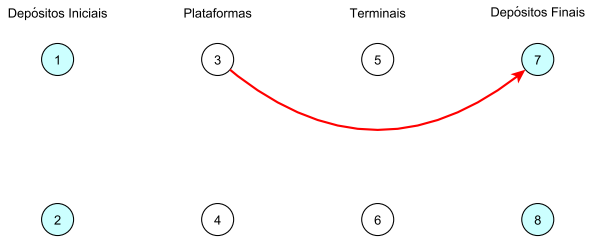

Figura 18: Exemplo da fixação de variável.

6. Não existe arco de um nó para ele mesmo:

   $x_{ijk} = 0, \ \forall i \in N, j \in N, k \in K$ se $i = j$;

7. Se um navio vindo de i não consegue chegar em j a tempo, então nunca ocorre esta visita:

   $x_{ijk} = 0, \ \forall i \in N, j \in N, k \in K$ se $e_i + d_i + t_{ij} > l_j$ e $dist_{ij} \neq 0$;

8. Essa fixação garante que para a mesma plataforma não existe arco que liga a segunda coleta à primeira coleta, se não existir sobreposição entre as duas janelas de tempo, ou seja, se não houver sobreposição das janelas de tempo de i e j, não há como o navio realizar as duas coletas consecutivamente:

$x_{ijk} = 0, \ \forall i \in P, j \in P, k \in K, dist_{ij} = 0$ e $e_i > l_j$;

9. Essa fixação garante que para o mesmo terminal não existe arco que liga a segunda entrega à primeira coleta se não existir sobreposição entre as duas janelas de tempo:

   $x_{ijk} = 0, \ \forall i \in D, j \in D, k \in K, dist_{ij} = 0$ e $e_i > l_j$;

10. Os nós i e j representam o mesmo terminal, porém são representados por dois nós diferentes. Sendo assim, se o início da janela de tempo do nó i for maior que o início da janela de tempo do nó j, não existe arco de i para j:

    $x_{ijk} = 0, \ \forall i \in D, j \in D, k \in K, dist_{ij} = 0$ e $e_i > e_j$;

11. Essas duas fixações garantem que não existe o arco de volta entre a entrega e a respectiva coleta:

    $x_{ijk} = 0, \ \forall i \in D, j \in P, k \in K, dist_{i,j+n} = 0$ e $e_i > l_{j+n}$;

    $x_{ijk} = 0, \ \forall i \in P, j \in D, k \in K, dist_{i,j-n} = 0$ e $e_j > l_{i+n}$;

12. Estas duas fixações garantem que o navio começa e termina sua rota vazio:

    $Q_{ik} = 0, \ \forall i \in ST, k \in K$;

    $Q_{ik} = 0, \ \forall i \in EN, k \in K$;

13. Se a capacidade do navio for menor que a quantidade demandada, então este não pode coletar, pois não possui a capacidade necessária:

    $Q_{ik} = 0, \ \forall i \in N, j \in N, k \in K$ se $Cap_k < |q_i|$;

14. Estas duas fixações determinam que x deve ser igual a 0 se a capacidade do navio é menor que a demanda do nó:

    $x_{ijk} = 0, \ \forall j \in P \cup D, i \in N, k \in K$ e $|q_j| > Cap_k$;

    $x_{ijk} = 0, \ \forall i \in P \cup D, j \in N, k \in K$ e $|q_i| > Cap_k$;

15. O navio sai exatamente no início da janela de tempo do seu depósito inicial específico:

    $B_{ik} = e_i, \ \forall i \in ST, k \in K$ e $i = s_k$;

16. Estas duas fixações garantem que para plataformas e terminais, se a demanda das duas plataformas, por exemplo, exceder a capacidade do navio, este não pode coletar na plataforma i e em seguida na plataforma j:

    $x_{ijk} = 0, \ \forall i \in D, j \in D, k \in K$ e $(-1)(q_i + q_j) > Cap_k$;

    $x_{ijk} = 0, \ \forall i \in P, j \in P$ e $q_i + q_j > Cap_k$;

17. Estas duas fixações garantem que o navio não sai de seu depósito inicial (final) e vai para o depósito inicial (final) de outro navio:

    $x_{ijk} = 0, \ \forall i \in ST, j \in ST, k \in K;$

    $x_{ijk} = 0, \ \forall i \in EN, j \in EN, k \in K;$

18. O navio não sai de seu depósito inicial e vai para o depósito final de outro navio:

    $x_{ijk} = 0, \ \forall i \in ST, j \in EN, k \in K$ e $j \neq (i + 2n + |K|);$

19. O depósito de chegada não pode receber os navios que não vão estacionar nele:

    $x_{ijk} = 0, \ \forall i \in P, j \in EN, k \in K$ e $k \neq (i - 2n - |K|)$ e $i \neq (j - 2n - |K|);$

20. Nenhum navio entra no nó s_k e isto é ilustrado pela Figura 19 em que não pode-se ter o arco (3,1) já que 3 é uma coleta e 1 é um depósito artificial inicial.

    $x_{ijk} = 0, \ \forall j \in ST, i \in N, k \in K;$

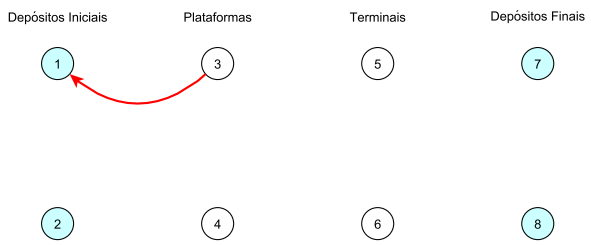

Figura 19: Exemplo da fixação de variável.

21. Nenhum navio sai do nó en_k:

    $x_{ijk} = 0, \ \forall i \in EN, j \in N, k \in K;$

22. Fixação quanto ao posicionamento dinâmico:

    $x_{ijk} = 0, \ \forall j \in C_{DP}, i \in N, k \notin K_{DP};$

23. Restrição de calado flexível escrita como fixação de variáveis:

    $x_{ijk} = 0, \quad \forall i \in N, j \in N, k \in K$ se $A_{ik} = 1$ e $CF_{ik} = 0;$

24. Se a janela de tempo do nó i fecha antes da abertura da janela de tempo do nó j, então não existe arco de j para i:

    $x_{jik} = 0$, se $i \in P \cup D, j \in P \cup D, k \in K$ e $e_j \geq l_i$.

Em Cordeau (2006) e Dumas et al. (1991), os autores mostram uma série de cortes que podem ser colocados diretamente no modelo para que melhorem seu limitante inferior. Os cortes são feitos para um modelo de coleta e entrega com janelas de tempo e frota heterogênea. Entretanto, os autores utilizam um artifício para não considerar o índice k dos navios e assim, gerar menos restrições. Este artifício consiste em definir a variável $x_{ij} = \sum_{k \in K} x_{ijk}$, e também é útil para o presente trabalho. A partir de agora, considere que o nó s_0 corresponde ao depósito inicial comum a todos os navios, e o nó en_0 corresponde ao depósito final comum a todos os navios. Os cortes abaixo foram exemplificados no capítulo anterior na parte relacionada ao método *branch-and-cut* (Seção 3.2.1), e são aqui aplicados para melhorar a relaxação linear do modelo. Para cada conjunto de nós S especificado abaixo tem-se um corte que é incluído diretamente no modelo matemático:

1. Eliminação sub-rota:

   - Para cada par de nós $i,j \in P$ os seguintes conjuntos S podem ser incorporados na restrição (3.17):
     - $S = \{i,j\} \Rightarrow \sum_{k \in K} (x_{ijk} + x_{jik} + x_{n+i,j,k} + x_{n+j,i,k}) \leq 1$;
     - $S = \{i,n+j\} \Rightarrow \sum_{k \in K} (x_{i,n+j,k} + x_{n+j,i,k} + x_{n+i,n+j,k}) \leq 1$;
     - $S = \{i,n+i,j\} \Rightarrow \sum_{k \in K} (x_{ijk} + x_{jik} + x_{i,n+i,k} + x_{j,n+i,k} + x_{n+i,j,k} + x_{n+j,i,k} + x_{n+j,n+i,k}) \leq 2$.
   - Para cada par de nós $i,j \in P$ os seguintes conjuntos S podem ser incorporados na restrição (3.18):
     - $S = \{n+i,n+j\} \Rightarrow \sum_{k \in K} (x_{n+i,n+j,k} + x_{n+j,n+i,k} + x_{n+i,j,k} + x_{n+j,i,k}) \leq 1$;
     - $S = \{i,n+j\} \Rightarrow \sum_{k \in K} (x_{i,n+j,k} + x_{n+j,i,k} + x_{i,j,k}) \leq 1$;
     - $S = \{i,n+i,n+j\} \Rightarrow \sum_{k \in K} (x_{i,n+j,k} + x_{n+j,i,k} + x_{i,n+i,k} + x_{n+j,n+i,k} + x_{n+i,n+j,k} + x_{ijk} + x_{n+i,j,k}) \leq 2$.
   - Similarmente, as inequações (3.19) e (3.20) resultam em:
     - $S = \{n+i,j,i\} \Rightarrow \sum_{k \in K} (x_{n+i,j,k} + 2x_{j,n+i,k} + x_{j,i,k} + x_{i,n+i,k} + x_{n+j,n+i,k}) \leq 2$;
     - $S = \{i,n+i,n+j\} \Rightarrow \sum_{k \in K} (x_{i,n+i,k} + x_{n+i,n+j,k} + x_{n+j,i,k} + 2x_{i,n+j,k} + x_{i,j,k}) \leq 2$.

- Para a inequação clássica de eliminação de sub-rota (3.16) temos o seguinte conjunto S e a seguinte inequação:
  - $S = \{i,j,n+i,n+j\} \Rightarrow \sum_{k \in K} (x_{i,j,k} + x_{i,n+i,k} + x_{i,n+j,k} + x_{j,i,k} + x_{j,n+i,k} + x_{j,n+j,k} + x_{n+i,i,k} + x_{n+i,j,k} + x_{n+i,n+j,k} + x_{n+j,i,k} + x_{n+j,j,k} + x_{n+j,n+i,k}) \leq 3.$

2. Restrição de Precedência (2.18): para cada par de nós $i,j \in P$ os seguintes casos particulares valem:
   - $S = \{s_0,i,n+j\} \Rightarrow \sum_{k \in K} (x_{s_0,i,k} + x_{i,n+j,k} + x_{n+j,i,k}) \leq 1;$
   - $S = \{i,n+j,en_0\} \Rightarrow \sum_{k \in K} (x_{i,n+j,k} + x_{n+j,i,k} + x_{n+j,en_0,k}) \leq 1;$
   - $S = \{s_0,i,n+i,n+j\} \Rightarrow \sum_{k \in K} (x_{s_0,i,k} + x_{i,n+i,k} + x_{i,n+j,k} + x_{n+j,i,k} + x_{n+i,n+j,k} + x_{n+j,n+i,k}) \leq 2;$
   - $S = \{i,j,n+j,en_0\} \Rightarrow \sum_{k \in K} (x_{i,j,k} + x_{j,i,k} + x_{i,n+j,k} + x_{n+j,i,k} + x_{j,n+j,k} + x_{n+j,en_0,k}) \leq 2.$

3. Restrições de ordem generalizadas (3.21): para cada par de coleta $i,j \in P$ a seguinte inequação é válida:

$$\sum_{k \in K} (x_{i,n+j,k} + x_{n+j,i,k} + x_{n+i,j,k} + x_{j,n+i,k}) \leq 1;$$

4. Restrição de Caminhos Infactíveis (3.26): para cada par de coletas $i,j \in P$ e $\forall k \in K$ tal que $t_{ij} + d_j + t_{j,n+j} + d_{n+j} + t_{n+j,n+i} > l_{en_k}$, sendo l_{en_k} o fim da janela de tempo do depósito final do navio k, então:

$$\sum_{k \in K} (x_{ijk} + x_{j,n+j,k} + x_{n+j,n+i,k}) \leq 1.$$

Além destes cortes propostos na literatura, propomos outros dois cortes com o intuito de melhorar o limitante inferior do modelo matemático:

1. Garante que a capacidade do navio seja respeitada:

$$\sum_{i \in P \cup D} \sum_{j \in P \cup D} q_i x_{ijk} \leq Cap_k, \quad \forall k \in K;$$

2. Garante que o instante de início de serviço no nó de coleta i deve ser menor ou igual ao instante de início de serviço no nó de entrega $n + i$:

$$B_{ik} + d_i + t_{i,i+n} \leq B_{i+n,k}, \ \ \forall i \in P, k \in K.$$

4.5 Exemplares de Teste

Nesta seção são apresentados os casos de teste utilizados para validação do modelo matemático proposto acima. Depois, são apresentados os casos de teste reais que foram disponibilizados pela empresa do estudo de caso para a realização de testes computacionais e por fim, os resultados computacionais do modelo matemático com os exemplares reais.

4.5.1 *Toy Models*

Realizamos inicialmente 12 casos teste contemplando diferentes possibilidades como, por exemplo, restrições quanto à atracação de navios e restrições quanto ao posicionamento dinâmico, com o objetivo de testar a consistência da modelagem. O termo *toy models* refere-se a exemplares de testes pequenos para que o modelo e sua solução possam ser analisados em detalhes. O modelo foi implementado na linguagem de modelagem *OPL* e resolvido com o *software* de otimização IBM CPLEX versão 12.5.1 (ILOG, 2013). O computador utilizado foi um PC Core *i*7 3.40 GHz, 16 GB de memória RAM e sistema operacional Windows 7 *Professional*. Em todos os exemplares de teste, o CPLEX é capaz de encontrar a solução ótima em segundos. Os casos teste são descritos a seguir, de acordo com o número de navios, plataformas e terminais, bem como as configurações de atracação e posicionamento dinâmico. A distância entre os pontos operacionais variam entre [0,10] unidades, a demanda dos nós variam entre 10 e 60 e a duração de todos os serviços é de 5 unidades de tempo. As janelas de tempo possuem variação de [0,200] e a capacidade dos navios varia de 100 a 130 unidades. Estes casos teste foram definidos de maneira empírica, ou seja, apoiados em todos os testes feitos no decorrer do desenvolvimento desta tese.

1. Dois navios, duas plataformas e dois terminais, sem restrições quanto à atracação, posicionamento dinâmico ou calado flexível;

2. Dois navios, duas plataformas e dois terminais. Além disso, o navio 2 não pode atracar na plataforma 4;

3. Dois navios, três plataformas e três terminais (o tamanho do problema aumentou em relação ao exemplar anterior). O navio 2 não pode atracar na plataforma 4;

4. Dois navios, três plataformas e três terminais. O navio 1 não pode atracar nas plataformas 3 e 5;

5. Dois navios, três plataformas e três terminais. Nenhum dos dois navios pode atracar no terminal 7, entretanto o calado do navio 1 é flexibilizado podendo assim, entrar no terminal 7 mediante a uma certa porcentagem de carga a bordo;

6. Dois navios, três plataformas e três terminais. Mesmo caso dado no item anterior com o calado flexível e além disso, o navio 1 possui posicionamento dinâmico;

7. Dois navios, três plataformas e três terminais em que o navio 1 também possui calado flexível, além deste possuir posicionamento dinâmico. Na plataforma 3 pode atracar somente navios que possuam DP;

8. Dois navios, três plataformas e três terminais. Aborda a questão do calado flexível para o navio 1 e este possui posicionamento dinâmico. As plataformas 3 e 5 aceitam somente navios com DP;

9. Três navios, quatro plataformas e quatro terminais. Os navios 1 e 2 não podem atracar na plataforma 7. Os navios 2 e 3 possuem posicionamento dinâmico e a plataforma 5 aceita somente navios com DP;

10. Três navios, quatro plataformas e quatro terminais. Este é o mesmo exemplar anterior, porém com custo de *stand-by* do navio 2 dez vezes maior que dos outros navios;

11. Três navios, quatro plataformas e quatro terminais. O navio 3 não pode atracar na plataforma 6. A plataforma 4 aceita somente navio com posicionamento dinâmico e o navio 3 possui DP;

12. Três navios, quatro plataformas e quatro terminais. Este exemplar é o mesmo que o anterior, exceto que nenhum navio possui posicionamento dinâmico, então o problema se torna infactível, já que nenhum navio pode atracar na plataforma 3.

As Tabelas 1 e 2 representam em síntese o que acontece em todos os casos teste. Para a Tabela 1, cada linha representa um caso teste diferente, enquanto que cada coluna representa o número de navios, número de plataformas/terminais, quais navios possuem

posicionamento dinâmico, impossibilidade de atracar e calado flexível. Para a Tabela 2 cada coluna representa a menor janela inferior, maior janela inferior, média das janelas inferiores, menor janela superior, maior janela superior e média das janelas superiores.

Tabela 1: Principais características dos *toy models*.

Exem.	# Navios	# Plataformas\ Terminais	Posicionamento Dinâmico	Impossibilidade de Atracar	Calado Flexível
1	2	2	-	-	-
2	2	2	-	navio 2 $\rightarrow$ plat. 4	-
3	2	3	-	navio 2 $\rightarrow$ plat. 4	-
4	2	3	-	navio 2 $\rightarrow$ plat. 3,5	-
5	2	3	-	navios 1,2 $\rightarrow$ term. 7	navio 1
6	2	3	navio 1	navios 1,2 $\rightarrow$ term. 7	navio 1
7	2	3	navio 1	-	navio 1
8	2	3	navio 1	-	navio 1
9	3	4	navios 2,3	navios 1,2 $\rightarrow$ plat. 7	-
10	3	4	navios 2,3	navios 1,2 $\rightarrow$ plat. 7	-
11	3	4	navio 3	navio 3 $\rightarrow$ plat. 6	-
12	3	4	-	navio 3 $\rightarrow$ plat. 6	-

Tabela 2: Principais características dos *toy models*.

Exem.	Menor Jan. Inferior	Maior Jan. Inferior	Média Jan. Inferior	Menor Jan. Superior	Maior Jan. Superior	Média Jan. Superior
1	10	60	32,5	40	100	75,0
2	10	60	32,5	40	100	75,0
3	10	60	33,3	40	100	78,3
4	10	60	33,3	40	100	78,3
5	10	60	33,3	40	100	78,3
6	10	60	33,3	40	100	78,3
7	10	60	33,3	40	100	78,3
8	10	60	33,3	40	100	78,3
9	20	80	47,5	40	120	83,7
10	20	80	47,5	40	120	83,7
11	20	80	47,5	40	120	83,7
12	20	80	47,5	40	120	83,7

Nas figuras a seguir tem-se a representação gráfica das soluções dos *toy models* para o problema de coleta e entrega e janelas de tempo na indústria petrolífera. Para os exemplos 1 e 2 os nós 1 e 2 são depósitos iniciais dos navios 1 e 2, respectivamente. Os nós 7 e 8 são os depósitos finais dos navios 1 e 2, respectivamente. Os nós 3 e 4 são as plataformas, que possuem uma oferta de produto e os nós 5 e 6 são os terminais, com demanda pareada com as plataformas. Para os exemplos de 3 a 8 os nós 1 e 2 são os depósitos de início dos navios 1 e 2 respectivamente. Os nós 9 e 10 são os depósitos de chegada dos navios 1 e 2, respectivamente. Os nós 3, 4 e 5 representam as plataformas e 6, 7 e 8 representam os terminais. Para os *toys* 9 a 12 os nós artificiais de saída dos

navios são 1, 2 e 3, respectivamente para cada navio. Os nós de chegada dos nós são 12, 13 e 14. As plataformas são indicadas pelos nós 4, 5, 6 e 7 e os terminais são indicados pelos nós 8, 9, 10 e 11. Cada navio inicia-se no seu depósito inicial de acordo com seu número, isto é, o navio 1 inicia no depósito inicial 1, o navio 2 no depósito inicial 2, e assim sucessivamente.

A Figura 20 mostra o caso *default* (caso 1), ou seja, em que não há restrições de atracação e nem restrições quanto ao posicionamento dinâmico. O caso 2 é o mesmo que o primeiro exceto que o navio 2 não pode atracar na plataforma 4 e pela Figura 21 vemos que é exatamente isto que ocorre. No *toy* 3 aumentamos um pouco o tamanho do problema e restringimos que o navio 2 novamente não pode coletar na plataforma 4. Isto é demonstrado pela Figura 22 em que o navio 2 não coleta em nenhuma plataforma. A Figura 23 ilustra o que acontece no quarto caso: o navio 1 não pode atracar nas plataformas 3 e 5, ou seja, estas plataformas são visitadas pelo navio 2. O quinto caso é utilizado para analisar a questão do calado flexível: nenhum dos dois navios podem atracar no terminal 7, porém se o calado do navio 1 é flexibilizado, então este passa a atracar no terminal 7, como é mostrado pela Figura 24.

O próximo caso é o mesmo que o anterior adicionando-se a seguinte premissa: que o navio 1 possui posicionamento dinâmico e, assim, pode atracar em qualquer plataforma. Isto é ilustrado na Figura 25. O caso 7 possui calado flexível para o navio 1, a plataforma 3 somente aceita navios com posicionamento dinâmico e o navio 1 possui DP. Sendo assim, a única solução factível é o navio 1 passar pelo terminal 7 e também na plataforma 3 (Figura 26). O caso 8 também aborda a questão do calado flexível no navio 1, além disso, o navio 2 possui posicionamento dinâmico e as plataformas 3 e 5 aceitam somente navios com DP. Sendo assim, a Figura 27 demonstra que o terminal 7 é visitado pelo navio 1 e as plataformas 3 e 5 pelo navio 2, conforme as premissas.

No próximo caso teste aumenta-se o tamanho do problema novamente. As premissas são respeitadas e isto é demonstrado na Figura 28 já que a plataforma 7 é visitada pelo navio 3 e plataforma 5 é visitada pelo navio 2, que possui posicionamento dinâmico. O caso 10 é o mesmo exemplar que o anterior, porém com o custo do navio 2 mais elevado. Isto faz com que o navio 2 não realize nenhuma coleta (Figura 29). Os últimos casos (11 e 12) abordam uma questão importante: a plataforma 4 aceita somente navios com posicionamento dinâmico, então no caso 11 assumimos que o navio 3 possui DP, este realiza a coleta na plataforma 4 e encontramos a solução ótima que é representada pela Figura 30. Entretanto, para o caso 12 admitimos que nenhum navio possui posicionamento

dinâmico. Isto fez com que o problema se tornasse infactível, já que a plataforma 4 aceita somente navios com DP.

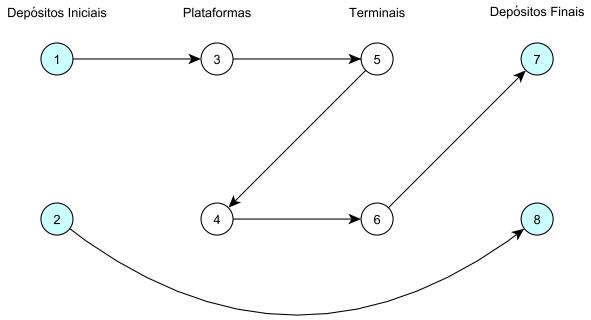

Figura 20: Representação gráfica para o *toy model* 1.

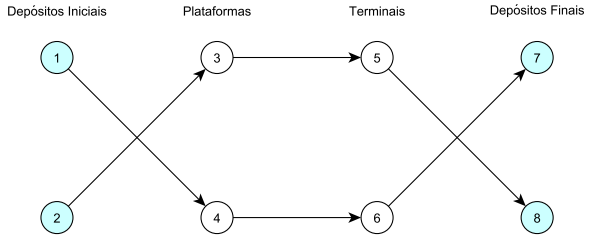

Figura 21: Representação gráfica para o *toy model* 2.

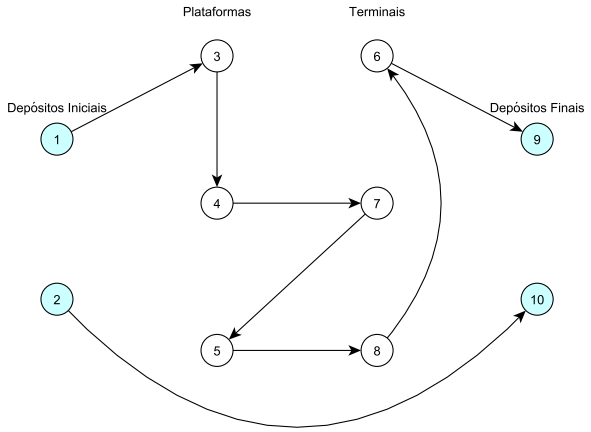

Figura 22: Representação gráfica para o *toy model* 3.

4.5.2 Definição dos exemplares reais de teste

Os casos utilizados para a realização dos testes computacionais foram disponibilizados pela empresa do estudo de caso e divididos em dois casos testes: caso 1 e caso 2. O primeiro

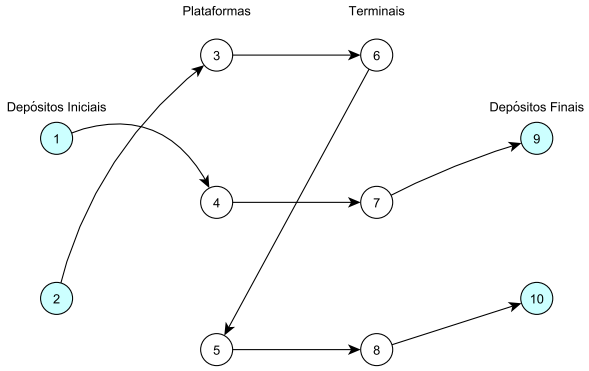

Figura 23: Representação gráfica para o *toy model* 4.

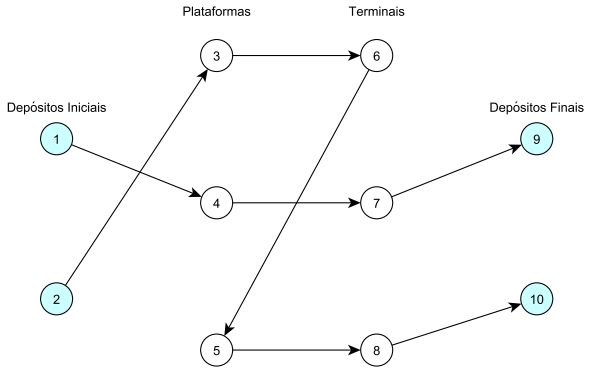

Figura 24: Representação gráfica para o *toy model* 5.

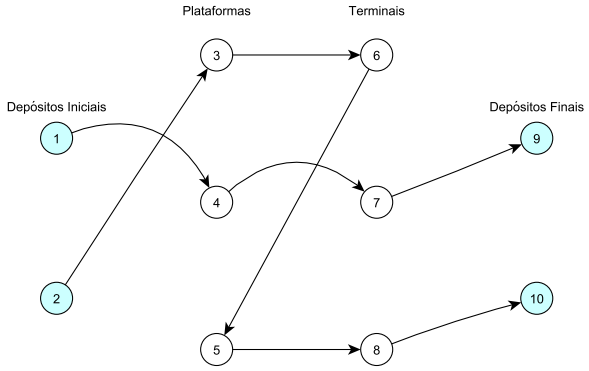

Figura 25: Representação gráfica para o *toy model* 6.

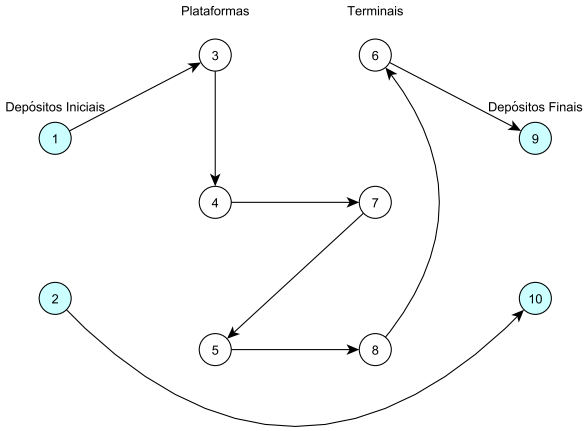

Figura 26: Representação gráfica para o *toy model* 7.

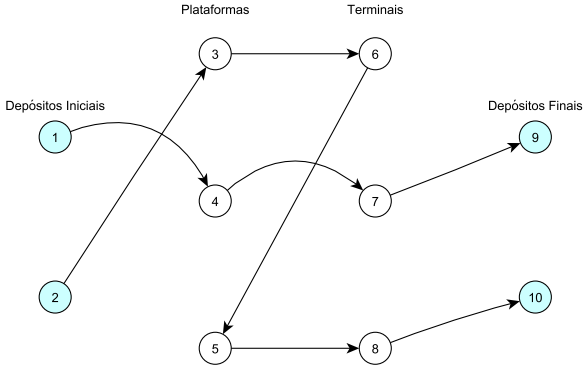

Figura 27: Representação gráfica para o *toy model* 8.

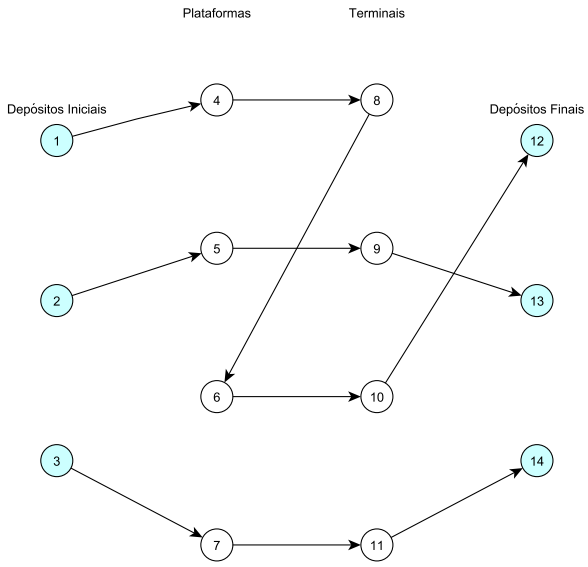

Figura 28: Representação gráfica para o *toy model* 9.

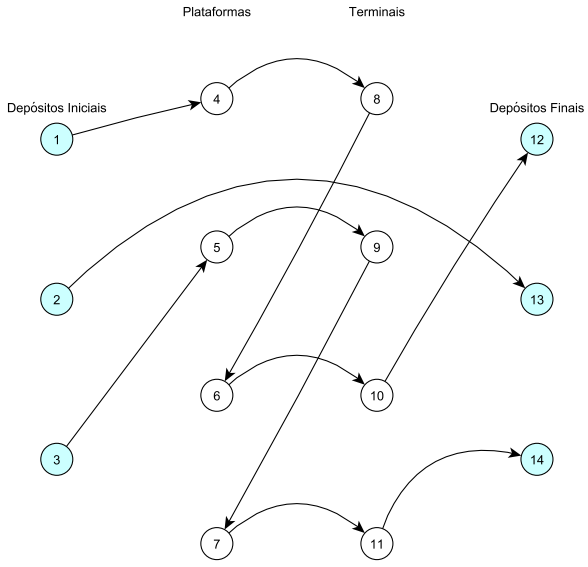

Figura 29: Representação gráfica para o *toy model* 10.

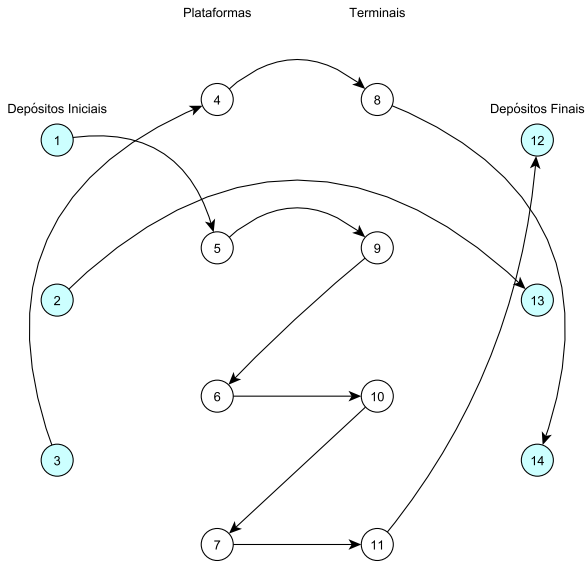

Figura 30: Representação gráfica para o *toy model* 11.

caso teste é referente à produção de petróleo no mês de Julho de 2013 e o segundo caso teste é referente à produção no mês de Janeiro de 2013.

As definições das janelas de tempo dos exemplares reais são baseadas em uma informação importante: as datas em que as plataformas vão alcançar seu nível máximo de estoque (na linguagem usualmente utilizada pelos operadores, "a data em que as plataformas vão chegar ao *top*"). Esta informação é processada e dada por um planejamento na empresa em nível hierárquico acima do roteamento e programação dos navios aqui estudado. Desta maneira, as janelas de tempo são geradas a partir das datas dos *tops* das plataformas. A janela de tempo inferior das plataformas ocorre no instante de tempo em que a plataforma já possui em estoque a quantidade demandada; a janela de tempo superior das plataformas é datada de 48 horas antes do instante de tempo em que a plataforma vai atingir seu nível máximo, ou seja, 48 horas antes de chegar ao *top*. O início da janela de tempo do terminal respectivo se dá no instante de tempo t, no qual t é determinado somando-se o início da janela de tempo da plataforma, o tempo de viagem entre a plataforma e o terminal e a duração de serviço da plataforma. O fim desta janela de tempo ocorre de maneira análoga, sendo determinado somando-se o fim da janela de tempo da plataforma, o tempo de viagem da plataforma para o terminal e a duração de serviço na plataforma. O horizonte de planejamento inicia-se 48 horas antes da primeira abertura da janela de tempo das plataformas e termina 49 horas após o fim do último fechamento da janela de tempo dos terminais.

Estes dois casos teste possuem custos relacionados ao navio se encontrar parado ou em movimentação bem como custos de atracação. Além disso, estes casos não possuem os pontos operacionais referentes às regiões Norte e Nordeste da costa brasileira (estas regiões são bastante específicas e, portanto, são tratadas separadamente pela empresa). Outra particularidade é que a matriz de distância entre os pontos operacionais, disponibilizada pela empresa do estudo de caso, teve de ser ajustada com o apoio do pessoal da empresa para que a desigualdade triangular fosse respeitada. Estes casos também contêm informações referentes ao posicionamento dinâmico, calado flexível e impossibilidade de atracação. Conforme mencionamos antes, são consideradas velocidades médias iguais para todos os navios. Além disso, define-se α_1 equivale a 0,3, $\alpha_2 = 0{,}5$ e $\beta = 1000$. Cabe salientar que o parâmetro M foi apertado em alguns experimentos preliminares, mas tais experimentos não mostraram melhorias substanciais, então por simplicidade adotamos o mesmo M para todos os exemplares.

O caso 1 conta com 25 navios disponíveis e um total de 142 pares de coleta e entrega durante o mês de Julho. O caso 2 conta com 31 navios disponíveis e um total de 83 pares de coleta e entrega referentes ao mês de Janeiro de 2013. Estes casos são subdivididos em outros exemplares menores para a realização dos experimentos conforme descrito na próxima seção. As próximas tabelas (Tabela 3, 4 e 5) descrevem em detalhes todas as características de cada caso e de cada exemplar de teste. Para ter ideia do tamanho das janelas de tempo, considere o exemplar $C2N25$ a maior abertura de janela de tempo é de 144 (diferença entre a janela superior e inferior) e a menor é 0, ou seja, o navio é obrigado a chegar e sair do ponto operacional exatamente no instante de abertura da janela de tempo.

Tabela 3: Descrição de cada caso.

Exemplar	#Veículos	Menor Capacidade	Maior Capacidade	Média Capacidade
Caso 1	25	50714	173738	129629
Caso 2	31	50714	176118	131780

4.6 Experimentos Computacionais

O número de coletas e entregas é subdivido em outros exemplares de testes para fins de experimentos com o modelo. Os exemplares que seguem possuem o nome $CxNy$, em que x indica o caso que o exemplar originou e y indica o número de pares de coleta e entrega.

Tabela 4: Descrição dos dados para o Caso 1.

Exem.	Menor Demanda	Maior Demanda	Média Demanda	Menor Janela Inferior	Maior Janela Inferior	Média Janela Inferior	Menor Janela Superior	Maior Janela Superior	Média Janela Superior
$C1N10$	10000	90000	39240,0	48,00	123,23	83,51	48	193,84	117,11
$C1N15$	10000	100000	52960,0	48,00	168,53	87,47	48	241,07	137,07
$C1N20$	7000	100000	56179,9	48,00	239,61	96,33	48	273,00	153,98
$C1N25$	7000	100000	50976,0	48,00	239,61	101,49	48	305,00	173,53
$C1N30$	7000	143090	57186,0	9,99	298,00	110,34	48	315,23	188,37
$C1N35$	5000	143090	54016,5	9,99	298,00	113,16	48	340,07	202,68
$C1N40$	5000	143090	51927,0	9,99	298,00	119,56	48	363,23	215,88
$C1N45$	4900	143090	52155,1	9,99	377,61	131,27	48	385,84	228,64
$C1N50$	4900	143090	52521,6	9,99	377,61	138,92	48	436,07	241,44
$C1N55$	4900	143090	52252,3	9,99	377,61	138,89	48	441,00	253,47
$C1N60$	4900	143090	51778,0	9,99	377,61	144,85	48	475,46	266,29
$C1N65$	4000	143090	51513,5	9,99	432,53	153,00	48	484,07	279,49
$C1N70$	4000	143090	50744,0	9,99	432,53	161,61	48	522,76	293,13
$C1N75$	3200	143090	49110,4	9,99	432,53	162,68	48	561,00	306,23
$C1N80$	3200	143090	50329,7	9,99	459,23	172,40	48	561,00	318,08

Tabela 5: Descrição dos dados para o Caso 2.

Exem.	Menor Demanda	Maior Demanda	Média Demanda	Menor Janela Inferior	Maior Janela Inferior	Média Janela Inferior	Menor Janela Superior	Maior Janela Superior	Média Janela Superior
$C2N10$	9500	80000	39980,0	47	138,76	83,65	48	190,38	122,15
$C2N15$	9500	90000	44266,7	47	150,53	85,57	48	246,53	141,64
$C2N20$	9500	90000	46200,0	47	258,76	99,60	48	282,76	159,66
$C2N25$	3500	90000	48100,0	47	264,53	114,54	48	312,07	176,98
$C2N30$	3500	90000	46230,0	47	264,53	113,19	48	346,69	193,23
$C2N35$	3500	90000	45625,7	47	288,76	121,98	48	346,69	208,41
$C2N40$	3500	90000	45672,5	47	313,84	136,13	48	402,76	224,95
$C2N45$	3500	90000	45408,9	47	336,53	147,29	48	432,53	242,25
$C2N50$	3500	90000	45668,0	47	449,76	161,22	48	459,23	259,66
$C2N55$	3500	90000	45152,7	47	449,76	168,09	48	486,30	276,34
$C2N60$	3500	90000	44148,3	47	454,38	185,35	48	526,38	292,99
$C2N65$	3500	90000	43136,9	47	478,38	191,43	48	609,00	311,46
$C2N70$	3500	90000	42560,0	47	478,38	196,18	48	818,53	337,13
$C2N75$	3500	90000	43144,0	47	694,69	212,87	48	1102,38	374,81
$C2N80$	3500	90000	41412,5	47	766,69	233,19	48	1202,54	421,32

Sendo assim, $C1N10$ indica 10 pares de coleta e entrega do Caso 1. Este exemplar é gerado a partir das 10 primeiras plataformas que dão *top* dentro do horizonte de planejamento, correspondendo a 10 pares de coleta nas plataformas e entrega nos terminais. Os testes com os Casos 1 e 2 foram realizados com o mesmo computador já especificado. O tempo limite para todos os casos teste foi de 5 horas de processamento, ou seja, 18000 segundos. Para ambos os casos, consideramos partes do horizonte de planejamento.

As Tabelas 6 e 7 mostram os resultados fornecidos pelo *software* de otimização CPLEX. A primeira coluna indica o nome do exemplar, isto é, seguindo a nomenclatura definida acima. A segunda coluna mostra o número de navios utilizados pela solução ótima ou sub-ótima, dependendo do *gap* de otimalidade. O valor da função objetivo é mostrado na terceira coluna denominada UB e a diferença relativa entre a melhor solução obtida e o limitante inferior (Gap) é dado em porcentagem ($Gap = \frac{\text{UB–LB}}{\text{UB}}$) e é apresentado na quarta coluna. Por fim, é mostrado o tempo computacional, dado em segundos para os exemplares de teste. As tabelas também mostram a diferença quando resolvemos o modelo matemático com e sem o pré-processamento discutido na Seção 4.4. O símbolo – indica que o tempo máximo permitido foi atingido e os espaçamentos em branco representam que o *solver* não encontrou solução factível no tempo máximo estipulado.

Tabela 6: Resultados computacionais para o Caso 1.

Exemplares	Sem Pré-Processamento				Com Pré-Processamento			
	# navios util.	UB	Gap (%)	Tempo (s)	# navios util.	UB	Gap (%)	Tempo (s)
$C1N10$	7	1677,84	0%	7,13	7	1677,84	0%	6,49
$C1N15$	10	2311,98	0%	38,67	10	2311,98	0%	20,62
$C1N20$	13	2748,36	24,14%	–	13	2748,36	0%	2191,53
$C1N25$					14	3694,58	61,04%	–
$C1N30$								
$C1N35$								
$C1N40$								
$C1N45$								
$C1N50$								

Os resultados computacionais para o primeiro caso real (Tabela 6) mostram que o pré-processamento realizado é importante para a solução do modelo. Para os dois primeiros exemplares o modelo sem o pré-processamento encontra solução ótima, porém com tempo em geral mais elevado, se comparado ao modelo com pré-processamento. Para o $C1N20$ o modelo sem o pré-processamento encontra a solução ótima, porém termina com *gap* de 24,14%, enquanto que o modelo com pré-processamento encontra e prova a solução ótima em 2191,53 segundos. Cabe ressaltar que para o exemplar $C1N25$ o modelo sem

Tabela 7: Resultados computacionais para o Caso 2.

Exemplares	Sem Pré-Processamento				Com Pré-Processamento			
	# navios util.	UB	Gap (%)	Tempo (s)	# navios util.	UB	Gap (%)	Tempo (s)
$C2N10$	7	1402,63	0%	23,05	7	1402,63	0%	16,34
$C2N15$	10	1708,51	47,34%	–	10	1708,51	17,13%	–
$C2N20$	12	2452,90	42,28%	–	12	2452,90	47,71%	–
$C2N25$					14	3300,28	59,84%	–
$C2N30$								
$C2N35$								
$C2N40$								
$C2N45$								
$C2N50$								

pré-processamento não encontrou solução factível no tempo estipulado. Os resultados com o pré-processamento mostram que para até 20 pares de coletas e entregas, o modelo resolve os exemplares otimamente, em que os exemplares $C1N10$ e $C1N15$ são resolvidos em poucos segundos. Para o exemplar $C1N25$ o *gap* apresentado é alto, perto de 60%, e o tempo computacional é elevado considerando que, para este exemplar, o *solver* parou por tempo limite de 5 horas de processamento. Entretanto, o *solver* ainda encontra solução inteira para este exemplar no tempo estipulado. A partir do $C1N30$, não é encontrado solução inteira em 5 horas de processamento.

A Tabela 7 mostra os resultados computacionais para o caso real 2. Como já esperado, o pré-processamento feito faz com que o *solver* encontre uma solução melhor que sem usar o pré-processamento, pois de fato melhora o limitante inferior e auxilia o *solver* a chegar mais rapidamente na solução ótima. Além disso, o pré-processamento faz com que o modelo reduza consideravelmente seu tamanho, pois vários arcos são previamente fixados em 0 e vários cortes são acrescentados ao modelo. Para o exemplar $C2N15$ resolvido pelo modelo sem o pré-processamento, o *gap* encontrado é alto; para $C2N20$ o modelo sem pré-processamento termina com *gap* um pouco mais baixo que o modelo com o pré-processamento. Em ambos os exemplares o *solver* para por limite de tempo. Para o modelo com pré-processamento, o exemplar $C2N10$ é encontrado a solução ótima em pouco tempo computacional enquanto que para os exemplares $C2N15$, $C2N20$ e $C2N25$, a solução ótima não é provada no tempo de 5 horas terminando com *gap*. Cabe ressaltar que para o exemplar $C2N25$ o *solver* somente encontrou solução factível no tempo estipulado, para o modelo com o pré-processamento. Para os demais exemplares, no tempo estipulado, não encontra-se nenhuma solução inteira factível dentro do limite de tempo de 5 horas.

Note que os valores da função objetivo apresentam custos relativamente baixos, uma

vez que o custo por visita consecutiva é de 1000 unidades e o valor da função objetivo não ultrapassa a casa das 4000 unidades. Isto ocorre, pois o custo fixo associado a cada navio não é contabilizado e, além disso, todos os custos foram reduzidos a casa das mil unidades. Para ter uma ideia do quanto vale o custo fixo, considere o $C1N25$, nele o custo fixo é de 4432,66, enquanto que a parte da função objetivo referente ao custo em movimento é de 1244,58 e a parte referente ao custo de atracação é de 2450,00. Neste exemplar, não se tem visitas consecutivas a plataformas diferentes.

Cabe dizer que a empresa resolve problemas comumente considerando apenas os pares de coletas e entregas previstas nas primeiras duas semanas, em vez de considerar os 142 (83) pares de coletas e entregas correspondentes ao mês de Julho (Janeiro) de 2013. Isto acontece pois são muitas as incertezas em relação aos demais pares, acarretando em problemas com cerca de 50 pares de coleta e entrega, em geral.

Com esses resultados computacionais podemos concluir que o modelo matemático, ainda que com todo o tratamento do pré-processamento, tem dificuldades de resolver exemplares de tamanhos mais realistas em tempos computacionais razoáveis com o *solver* CPLEX. Algum dos fatores associados podem ser o número de variáveis binárias, a relaxação linear fraca e simetria do problema. Para se ter uma ideia do tamanho do problema, considere o $C1N25$, o qual possui 286103 restrições e 270631 variáveis, enquanto que o $C1N50$ possui 851874 restrições e 632506 variáveis. Isto nos motiva a propor métodos específicos do tipo *branch-and-cut* para resolver o problema da indústria petrolífera, conforme próximo capítulo.

5 *Método* branch-and-cut *para o problema da indústria petrolífera*

Este capítulo tem o objetivo de apresentar o método *branch-and-cut* desenvolvido especificamente para resolver o problema de coleta e entrega com janelas de tempo da indústria petrolífera, descrito no Capítulo 4. São propostas algumas variações do método tendo como base modelos com variáveis de 2-índices e com variáveis de 3-índices. Primeiramente, são descritos os detalhes dos modelos utilizados. Em seguida, são explicados os algoritmos de separação para os cortes utilizados, seguidos de uma síntese de cada método proposto. Depois, é descrito o pré-processamento, por fim, os resultados computacionais são detalhados e analisados na última seção.

5.1 Modelagem Matemática

Neste capítulo são propostas diversas variações dos modelos e métodos do tipo *branch-and-cut*. Para facilitar a leitura, segue abaixo um esquema dos modelos propostos.

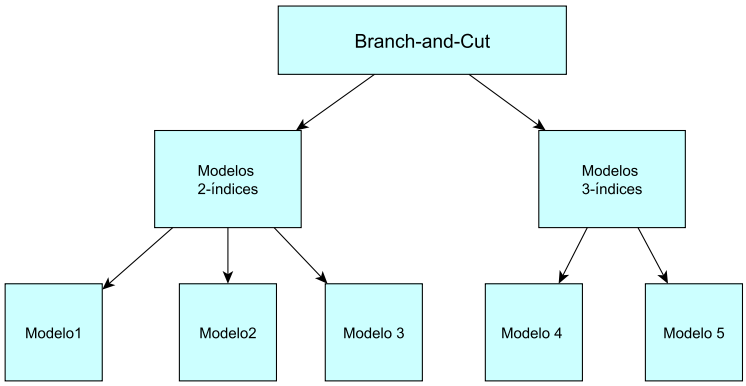

Figura 31: Esquema dos modelos propostos neste capítulo.

Os modelos estão descritos em detalhes nas próximas seções. Abaixo tem-se uma

breve explicação das características de cada modelo:

- O modelo 1 é um modelo de 2-índices baseado no modelo PDPTW2 (Seção 2.2.2), em que a frota de navios é homogênea (mas por meio de um artifício permite considerar certas heterogeneidades dos navios) e também considera as relações de precedência e restrições relacionadas à atracação, calado flexível e posicionamento dinâmico são dadas por meio de cortes. As outras características e imposições do problema são consideradas através de manipulação nos dados de entrada ou nas restrições;

- O modelo 2 é um modelo de 2-índices baseado no modelo PDPTW3 (Seção 2.2.2), com frota de navios homogênea (mas que também considera certas heterogeneidades dos navios por meio de artifícios) em que tem-se somente restrições de que todos os nós sejam visitados. As restrições de janelas de tempo, capacidade, relações de precedência, atracação, calado flexível e posicionamento dinâmico são dadas através de cortes;

- O modelo 3 também é um modelo de 2-índices com uma nova proposta para as restrições de precedência. Nesta proposta, a precedência entre os pares de coleta e entrega é garantida por um número polinomial de restrições, diferentemente dos modelos anteriores, nos quais o número de restrições de precedência é, teoricamente, exponencial em relação ao número total de nós no problema. Desta maneira, os únicos cortes obrigatórios são os referentes à atracação, calado flexível e posicionamento dinâmico;

- O modelo 4 é uma variação do modelo 3-índices apresentado no Capítulo 4. Este modelo é similar ao já apresentado, exceto que agora todos os navios iniciam sua rota num depósito comum inicial e terminam sua rota em um outro depósito comum a todos os navios. A motivação de colocar mais dois depósitos comuns a todos os navios neste modelo é para que os cortes propostos na literatura sejam válidos também para esta variação de modelagem;

- Por fim, o modelo 5 também é uma variação do modelo de 3-índices já apresentado. Neste caso, todos os navios também iniciam sua rota em um depósito comum e terminam sua rota em um depósito comum final a todos os navios, como no Modelo 4. Além disso, as restrições relacionadas à atracação, calado flexível e posicionamento dinâmico são a princípio eliminadas do modelo e, então, adicionadas por meio de cortes os quais se tornam agora obrigatórios para a garantia de factibilidade da solução ótima.

5.1.1 Modelo 1

Este primeiro modelo é baseado no modelo PDPTW2 descrito no Capítulo 2 e originalmente proposto por Ropke et al. (2007). Antes de apresentar o modelo cabe relembrar algumas notações. A frota de navios é homogênea, ou seja, possui navios idênticos com mesma capacidade Cap. As imposições de coleta e entrega juntamente com as relações de precedência são feitas através da definição do conjunto $\mathcal{S}$ de todos os subconjuntos de nós $S \subseteq N$ tal que $s_0 \in S$, $en_0 \notin S$, sendo que s_0 é o depósito inicial comum a todos os navios e en_0 é o depósito final comum a todos os navios, e existe pelo menos um pedido i tal que $i \notin S$ e $n + i \in S$.

Embora o modelo admita frota homogênea, é possível impor a heterogeneidade da frota em relação à capacidade, com um artifício nos dados de entrada do modelo. Inicialmente, Cap é definida como a capacidade do maior navio. O grafo é definido da seguinte maneira: temos o depósito s_0 comum a todos os navios. Os k primeiros nós para onde os navios seguem a partir do depósito s_0 representam a situação inicial de cada navio (depósito específico de cada navio). No depósito específico do navio k, uma carga artificial é coletada referente à diferença entre Cap e a capacidade específica do navio k. No depósito final específico do navio k, esta carga artificial é entregue antes do navio k ir para o depósito final comum a todos os navios (depósito en_0). Este truque nos permite que cada navio tenha capacidade diferente e inicie e termine sua rota em seu depósito específico, ao custo de $2\,|K|$ nós a mais definidos no grafo.

Para que o cálculo da função objetivo seja o mesmo que no modelo de 3-índices (Capítulo 4), utilizamos os mesmos parâmetros do modelo apresentado para o problema de coleta e entrega e janelas de tempo para a indústria petrolífera (Seção 4.3). Os custos relacionados à atracação e a penalização por visita consecutiva independem do navio utilizado. Entretanto, algumas adaptações são necessárias para que os custos relacionados ao consumo de cada navio sejam equivalentes ao do modelo com frota heterogênea. Na sequência, propõe-se como incorporar tal custo à função objetivo do modelo que é uma das contribuições desta tese.

Através da nova definição do grafo, sabemos que cada navio entrega a demanda artificial no seu depósito final específico, antes de chegar ao depósito comum a todos os navios, en_0. Ou seja, *a priori* temos conhecimento de que, se uma rota passa pelo nó j e este nó corresponde ao nó artificial final do navio k, então esta rota corresponde a rota do navio k. Através de uma nova variável contínua denominada R podemos calcular o tempo que cada navio se movimentou durante o horizonte de planejamento. Esta variável

irá guardar a informação do quanto o navio se movimentou durante o horizonte de tempo e com isto, conseguimos calcular seu valor na função objetivo. A Figura 32 exemplifica como a heterogeneidade em relação à capacidade é realizada e também como a diferença em relação aos custos dos navios é calculada na função objetivo. Na figura podemos ver que o navio 1 coleta a demanda artificial $Cap - Cap_1$ em seu depósito artificial inicial e a entrega em seu depósito artificial final específico (nó 9). Com esta informação a respeito do nó 9 (depósito final específico do navio 1) conseguimos calcular o custo do navio 1 em movimento durante o horizonte de planejamento, através da variável R_9. O mesmo pode ser feito para o navio 2.

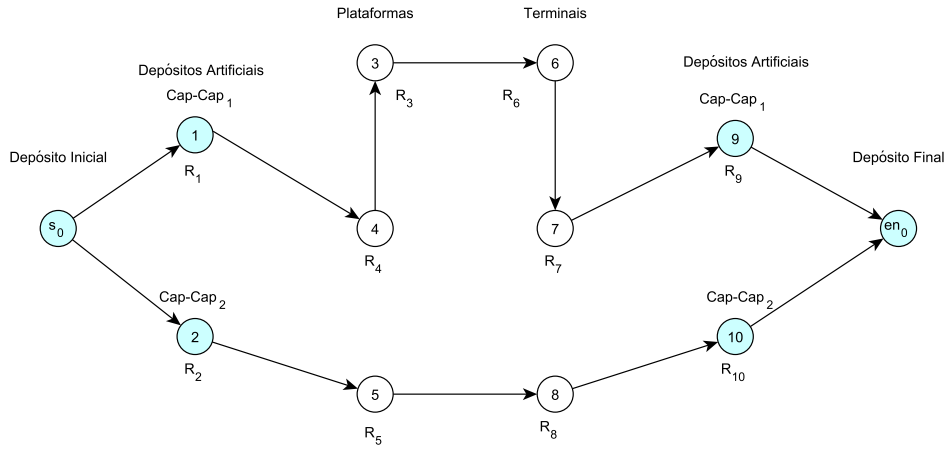

Figura 32: Exemplo de como a heterogeneidade dos navios é tratada em relação à capacidade e custos.

Agora, vamos relembrar as definições de conjuntos, parâmetros e variáveis dadas no Capítulo 4 e que também serão utilizadas neste capítulo:

Conjuntos

- $P = \{1,...,n\}$ é o conjunto dos nós das coletas;
- $D = \{n+1,...,2n\}$ é o conjunto dos nós das entregas;
- $N = P \cup D \cup \{s_0\} \cup \{en_0\}$, sendo que s_0 e en_0 são os depósitos inicial e final, respectivamente;
- ST é o conjunto dos nós artificiais iniciais de cada navio. $ST = \{s_1,s_2,...,s_k\}$ e $ST \subset P$;

- EN é o conjunto dos nós artificiais finais de cada navio. $EN = \{en_1,en_2,...,en_k\}$ e $EN \subset D$.

Parâmetros

- e_i corresponde ao início da janela de tempo no nó $i \in N$;
- l_i corresponde ao fim da janela de tempo no nó $i \in N$;
- q_i representa a demanda do nó $i \in P$. Esta demanda é entregue no nó $n+i$ e, assim, tem-se $q_{n+i} = q_i$;
- Cap é a capacidade do maior navio;
- cm_k é o consumo de combustível do navio k em movimento, $\forall k \in K$;
- cs_k é o consumo de combustível do navio k enquanto está parado (*stand-by*), $\forall k \in K$;
- $CustoTotal_k = cm_k - cs_k$ é o custo do navio k;
- ca_j é o custo por atracação no nó j;
- $dist_{ij}$ representa a distância do nó $i \in N$, para o nó $j \in N$;
- t_{ij} é o tempo de deslocamento do navio entre o nó $i \in N$ e o nó $j \in N$ (independente do navio k, pois estamos assumindo a mesma velocidade média para todos os navios);
- d_i é o tempo de serviço do nó $i \in N$;
- β é a penalização na função objetivo a visitas consecutivas a duas plataformas diferentes;
- M é um número suficientemente grande.

Variáveis

- x_{ij} é binária e assume valor 1 se algum navio percorre o arco $(i,j) \in N$, e 0 caso contrário;
- B_i é uma variável contínua e indica o instante de início de serviço do nó $i \in N$;
- Q_i é contínua e indica a quantidade de carga a bordo do navio após a visita em $i \in N$;

- VC_{ij} é uma variável inteira e assume valor positivo se algum navio visita a plataforma i e, em seguida, visita a plataforma j, com i e j sendo plataformas diferentes;

- R_i é uma variável contínua que indica o tempo total de movimento do navio até chegar ao nó $i \in N$.

Sendo assim, o Modelo 1 para o problema de coleta e entrega e janelas de tempo na indústria petrolífera é dado por:

$$\text{Min} \sum_{k \in K} CustoTotal_k R_{en_k} + \sum_{i \in N} \sum_{\substack{j \in P \cup D \\ dist_{ij} > 0}} ca_i x_{ij} + \sum_{i \in D} \sum_{j \in EN} ca_i x_{ij} + \sum_{i \in P} \sum_{j \in P} \beta * VC_{ij} \tag{5.1}$$

s.a

$$\sum_{i \in N} x_{ij} = 1 \qquad \forall j \in P \cup D \tag{5.2}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in P \cup D \tag{5.3}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 2 \qquad \forall S \in \mathcal{S} \tag{5.4}$$

$$B_j \geq (B_i + d_i + t_{ij})\, x_{ij} \qquad \forall i \in N; j \in N \tag{5.5}$$

$$Q_j \geq (Q_i + q_j)\, x_{ij} \qquad \forall i \in N; j \in N \tag{5.6}$$

$$e_i \leq B_i \leq l_i \qquad \forall i \in N \tag{5.7}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{Cap, Cap + q_i\} \qquad \forall i \in N \tag{5.8}$$

$$x_{ij} \leq VC_{ij} \qquad \forall i,j \in P : dist_{ij} > 0 \tag{5.9}$$

$$R_j \geq (R_i + t_{ij})\, x_{ij} \qquad \forall i,j \in N \tag{5.10}$$

$$\sum_{j \in (P \cup \{en_k\})} x_{s_k,j} = 1 \qquad \forall k \in K \tag{5.11}$$

$$\sum_{i \in (\{s_k\} \cup D)} x_{i,en_k} = 1 \qquad \forall k \in K \tag{5.12}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in N; j \in N \tag{5.13}$$

$$B_i, Q_i, VC_{i,j}, R_i \geq 0 \qquad \forall i,j \in N \tag{5.14}$$

A função objetivo (5.1) modela os custos relacionados ao consumo em movimento

de cada navio, custo de atracação e penaliza a visita consecutiva a duas plataformas diferentes, da mesma forma que o modelo do Capítulo 4. A imposição de que cada nó seja visitado exatamente uma única vez é dado pelas restrições (5.2) e (5.3). As restrições (5.4) são as restrições de precedência que garantem que a coleta de um dado nó i seja feita antes da sua entrega respectiva $(n+i)$; além disso, ambos os nós de coleta e entrega devem ser visitados pelo mesmo navio. Estas restrições já foram revisadas no Capítulo 2 e são iguais as restrições (2.18). A consistência das variáveis de tempo e quantidade de carga após a visita no nó são garantidas pelas restrições (5.5) e (5.6), respectivamente. Em (5.7) e (5.8) garantimos que as janelas de tempo e a capacidade do navio sejam respeitadas, respectivamente. As restrições (5.5) e (5.7) juntas garantem que não existem sub-rotas na solução ótima. As restrições (5.9) são utilizadas para contabilizar a visita consecutiva de um navio k a duas plataformas diferentes permitindo penalizar tais eventos na função objetivo, conforme sugestão dos operadores da empresa. As restrições (5.10) contabilizam o tempo em que o navio k se movimentou durante o horizonte de tempo. Por fim, as restrições (5.11) e (5.12) garantem que todos os navios saem dos nós artificiais de origem específicos e cheguem aos depósitos artificiais de destino específicos. A integralidade e domínio das variáveis são dadas por (5.13) e (5.14).

Note que as restrições (5.5), (5.6) e (5.10) são não-lineares, que podem ser linearizadas como:

- $B_j \geq B_i + d_i + t_{ij} - M\,(1 - x_{ij})\,;$
- $Q_j \geq Q_i + q_j - M\,(1 - x_{ij})\,;$
- $R_j \geq R_i + t_{ij}\,(x_{ij} - 1)\,M.$

Se enumerarmos todos os conjuntos $S \in \mathcal{S}$ das restrições (5.4), seria possível resolver esse modelo de forma direta usando-se um *software* de otimização (por exemplo IBM CPLEX), como foi feito com o modelo proposto no Capitulo 4. Entretanto, tem-se um número exponencial de restrições neste caso, da ordem de $2^{|N|}$, tornando esta abordagem inviável computacionalmente, em geral. Sendo assim, recorre-se a um método *branch-and-cut* tendo como base a relaxação combinatória deste conjunto de restrições. Note que a (5.4) é necessária para garantir a factibilidade da solução ótima. Sem ela, não teríamos a relação de precedência garantida, ou seja, que os nós de coleta sejam visitados antes das respectivas entregas e pelo mesmo navio. Portanto, neste modelo temos quase todas as restrições do modelo apresentado no Capítulo 4 exceto pela questão da precedência, atracamento, calado flexível e posicionamento dinâmico.

5.1.2 Modelo 2

Este modelo foi inspirado no modelo PDPTW3 descrito no Capítulo 2, originalmente proposto por Ropke et al. (2007). Para este modelo a frota de navios também é considerada homogênea. A heterogeneidade da frota em relação à capacidade é tratada pelo mesmo artifício descrito para o Modelo 1. Agora, além das relações de precedência, atracação, calado flexível e posicionamento dinâmico, as restrições de janela de tempo e capacidade também são dadas através de cortes, ou seja, para garantir a factibilidade da solução ótima, temos que analisar todas estas características sempre que uma solução inteira for encontrada.

Antes, porém, vamos relembrar algumas definições dos Capítulos 2 e 3 juntamente com algumas inequações. Optamos por reescrever certos trechos a fim de facilitar a leitura. Seja $\mathcal{R}$ o conjunto dos caminhos infactíveis em relação às janelas de tempo e seja $A(R)$ e $N(R)$ o conjunto de arcos e nós de um dado caminho $R \in \mathcal{R}$, respectivamente. Sendo assim, a inequação é dada por:

$$\sum_{(i,j)\in A(R)} x_{ij} \leq |A(R)| - 1 \quad \forall R \in \mathcal{R}. \tag{5.15}$$

Além disso, a restrição de capacidade é dada por:

$$\sum_{i,j\in S} x_{ij} \leq |S| - \max\left\{1, \lceil |q(S)| / Cap \rceil\right\}, \forall S \subseteq N \backslash \{s_0, en_0\}, |S| \geq 2. \tag{5.16}$$

Cabe ressaltar que este conjunto S difere da definição do conjunto S da restrição de precedência dada anteriormente, conforme explicitado em (5.16). Temos que, para qualquer $S \subset P \cup D$, $q(S) = \sum_{i\in S} q_i$, ou seja, somamos a demanda de todos os nós que estão no conjunto S. Sabemos *a priori* que a restrição de capacidade é dada otimamente se resolvermos o problema de empacotamento (*bin packing problem* - BPP). No caso geral, é inviável computacionalmente resolvermos otimamente este problema, então consideramos o seu limitante inferior, que é uma aproximação do resultado que seria obtido pelo problema de empacotamento (lado direito da inequação). Nesta aproximação, consideramos retirar um ou mais arcos de S, tomando o módulo de S subtraindo o max entre 1 e a demanda total de S dividida pela capacidade do veículo. O modelo 2 completo é dado por:

Min(5.1)

s.a

$(5.2) - (5.4)$

$(5.9) - (5.13)$

$$\sum_{(i,j)\in A(R)} x_{ij} \leq |A(R)| - 1 \qquad \forall R \in \mathcal{R} \tag{5.15}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - \max\{1, \lceil |q(S)| / Cap \rceil\} \qquad \forall S \subseteq N \setminus \{s_0, en_0\}; |S| \geq 2 \tag{5.16}$$

Neste modelo temos apenas as restrições de que todos os nós devem ser visitados. Todas as outras restrições do problema proposto no Capítulo 4 (janelas de tempo, capacidade, precedência, atracação, calado flexível e posicionamento dinâmico) são dadas através de cortes. Como é inviável enumerar todos os conjuntos S e R, utilizamos o método *branch-and-cut* juntamente com a relaxação combinatória.

5.1.3 Modelo 3

Este modelo também é um modelo de 2-índices com frota homogênea assim como os modelos 1 e 2, mas aqui se propõe uma modelagem alternativa para as restrições de precedência, que tem como vantagem um número polinomial de restrições (FURTADO et al., 2015). Sendo assim, temos um modelo baseado no modelo para o roteamento de navios com janelas de tempo clássico da literatura, no qual com o auxílio de uma nova variável contínua, impomos os pares de coleta e entrega. Neste modelo, a relaxação combinatória é aplicada apenas às restrições específicas do problema desse estudo (calado flexível, posicionamento dinâmico e atracação). As outras restrições estão explicitamente no modelo, ou estão contempladas na manipulação dos dados de entrada do modelo.

Através da variável B_i a qual indica o tempo de início de serviço no nó i, podemos satisfazer as relações de precedência entre os pares de coleta e entrega através das seguintes restrições:

$$B_{n+i} \geq B_i + d_i + t_{i,n+i}, \qquad \forall i \in P.$$

Sendo assim, temos apenas a garantia de que o nó de coleta é visitado antes do nó

de entrega. Agora, precisamos garantir que ambos os nós estejam na rota de um mesmo navio. Isto é garantido definindo um nova variável contínua para identificar as rotas, guardando o índice do primeiro nó visitado em cada rota. Sendo assim, os nós de coleta e entrega devem ter esse mesmo identificador para que estejam na rota de um mesmo navio. Formalmente temos que v_i é uma variável contínua que é igual ao índice do primeiro nó visitado na rota que visita o nó $i \in N$. Por exemplo, se j é o primeiro nó visitado na rota que visita i, então temos que $v_i = j$. Como consequência, temos que o nó de coleta i e o nó de entrega $n + i$ são visitados pelo mesmo navio se, e somente se, $v_i = v_{n+i}$. A Figura 33 ilustra os valores das variáveis v em um exemplo com três pares de coletas e entregas. Os nós 1, 2 e 3 são os nós de coleta e os nós 4, 5 e 6 são os nós das entregas respectivas. A primeira rota visita o nó 1 e com isto temos que $v_i = 1$ para todos os nós desta rota, ou seja, $v_1 = v_4 = 1$. Do mesmo modo tempos que o primeiro nó visitado da segunda rota é o nó 2, então temos que $v_i = 2$ para todos os nós desta rota, ou seja, $v_2 = v_3 = v_5 = v_6 = 2$. Neste contexto, é indiferente o valor das variáveis de v_{s_0} e v_{en_0}.

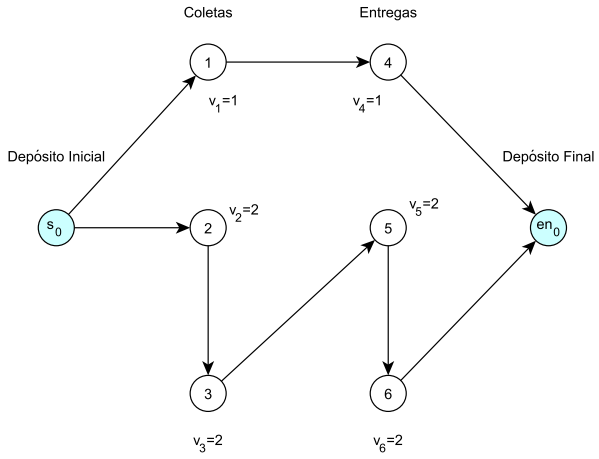

Figura 33: Valores das variáveis v_i em um exemplo com duas rotas.

Como já mencionado, para garantir que os nós de coleta e entrega estejam na mesma rota, temos as seguintes restrições:

$$v_i = v_{n+i}, \quad \forall i \in P.$$

Além disso, temos que garantir que se o nó j é o primeiro nó a ser visitado na rota de i, então o nó j será o índice que deve ser guardado. Isto é garantido pelas seguintes

restrições:

$$v_j \geq j \cdot x_{s_0 j}, \qquad \forall j \in P \cup D,$$
$$v_j \leq j \cdot x_{s_0 j} - n\left(x_{s_0 j} - 1\right), \qquad \forall j \in P \cup D.$$

Se o nó $j \in P \cup D$ é o primeiro nó visitado na rota, temos que $x_{s_0 j} = 1$ e com isto, as duas restrições juntas garantem que $v_j = j$. Por outro lado, se j não é o primeiro nó visitado na rota, então estas restrições ficam redundantes. Por último, precisamos garantir que todas as variáveis v de uma mesma rota, tenham o mesmo índice, ou seja, o índice do primeiro nó visitado. Isto é assegurado pelas seguintes restrições:

$$v_j \geq v_i + n\left(x_{ij} - 1\right) \qquad \forall i, j \in P \cup D,$$
$$v_j \leq v_i + n\left(1 - x_{ij}\right) \qquad \forall i, j \in P \cup D.$$

Desta maneira, se uma rota visita o nó i e viaja diretamente para o nó j, então $x_{ij} = 1$ e as restrições acima implicam que $v_j = v_i$. Caso contrário, temos que $x_{ij} = 0$, e as restrições se tornam redundantes. No caso específico do estudo de caso, o índice do primeiro nó visitado da rota é dado pelo nó da coleta artificial feita por cada navio. Sendo assim, temos que os índices guardados são os nós artificiais iniciais dos respectivos navios. O modelo 3 completo é dado por:

Min(5.1)

s.a

(5.2) – (5.3)

(5.5) – (5.14)

$$B_{n+i} \geq B_i + d_i + t_{i,i+n} \qquad \forall i \in P \qquad (5.17)$$
$$v_{n+i} = v_i \qquad \forall i \in P \qquad (5.18)$$
$$v_j \geq j \cdot x_{s_0 j} \qquad \forall j \in P \cup D \qquad (5.19)$$
$$v_j \leq j \cdot x_{s_0 j} - M\left(x_{s_0 j} - 1\right) \qquad \forall j \in P \cup D \qquad (5.20)$$
$$v_j \geq v_i + M\left(x_{ij} - 1\right) \qquad \forall i \in P \cup D \cup \{s_0\}\,; j \in P \cup D \cup \{en_0\} \qquad (5.21)$$
$$v_j \leq v_i + M\left(1 - x_{ij}\right) \qquad \forall i,j \in P \cup D \qquad (5.22)$$
$$v_{s_0} = 0 \qquad (5.23)$$

As restrições (5.17) garantem que o tempo de início de serviço do nó de entrega seja maior ou igual ao tempo de início de serviço do nó da coleta respectiva mais os tempos de serviço e viagem deste nó até o nó de entrega. Em (5.18) garante-se que o mesmo navio visita os nós de um par de coleta e entrega. As restrições (5.19) e (5.20) juntas impõem que o primeiro nó a ser visitado da rota é o índice guardado por toda a rota do navio específico. Em (5.21) e (5.22) garante-se que se ocorreu a visita ao nó i e logo após ao nó j, ambos os nós terão o mesmo índice. Por último, temos a restrição (5.23) que garante que o índice do depósito inicial s_0 é igual a 0.

Neste modelo 3 tem-se quase todas as restrições apresentadas no modelo proposto no Capítulo 4 exceto pelas restrições de atracação, calado flexível e posicionamento dinâmico.

5.1.4 Modelo 4

O modelo 4 é igual ao modelo de 3-índices apresentado no Capítulo 4, exceto que agora todos os navios saem de um depósito comum (depósito s_0) e retornam a um depósito comum (depósito en_0) a todos os navios no início e fim do horizonte de planejamento. Para que isto seja garantido 3 restrições são acrescentadas ao modelo do Capítulo 4, resultando-se no Modelo 4 que é dado por:

$$\text{Min}(4.1)$$

s.a

$$(4.2) - (4.26)$$

$$\sum_{j \in ST} x_{s_0 jk} = 1 \qquad \forall k \in K \tag{5.24}$$

$$\sum_{i \in EN} x_{i,en_0,k} = 1 \qquad \forall k \in K \tag{5.25}$$

$$\sum_{j \in N} x_{jik} - \sum_{j \in N} x_{ijk} = 1 \qquad \forall i \in P \cup D; k \in K \tag{5.26}$$

Estas restrições (5.24)-(5.26) são acrescentadas devido aos cortes que, por definição, consideram que todos os navios saem de um depósito comum (depósito s_0) e retornam a um depósito comum (depósito en_0) a todos os navios. É possível resolver este Modelo 4 utilizando um *solver* de propósito geral como, por exemplo, o CPLEX.

5.1.5 Modelo 5

O Modelo 5 é uma variação do modelo 4 em que utiliza-se da relaxação combinatória e retira-se as restrições de atracação, calado flexível e posicionamento dinâmico e coloca-se cortes para garantir a factibilidade da solução ótima em relação a estas restrições. Sendo assim, o Modelo 5 é dado por:

$$\text{Min} (4.1)$$

s.a

$$(4.2) - (4.17)$$

$$(4.22) - (4.26)$$

$$(5.24) - (5.26)$$

Portanto, este modelo difere do anterior apenas em relação às restrições de atracação, calado flexível e posicionamento dinâmico.

5.2 Algoritmos de Separação

Esta seção se inicia com os algoritmos de separação feitos exclusivamente para o problema de coleta e entrega e janelas de tempo na indústria petrolífera. Em seguida, são mostrados os algoritmos de separação para as restrições da literatura.

5.2.1 Caminhos Infactíveis - Estudo de Caso

A questão da impossibilidade de atracar, calado flexível e também posicionamento dinâmico são tratadas por uma mesma estratégia de separação: percorre-se cada caminho da solução do problema relaxado e, em cada um, é analisado se alguma destas restrições é violada, ou seja, uma forma de implementar é percorrer uma árvore de caminhos. Caso alguma restrição tenha sido violada, um corte correspondente é acrescentado ao problema, ou seja, se algum caminho for infactível na solução do problema relaxado em relação a atracação, calado flexível ou posicionamento dinâmico, então este caminho é proibido no problema por meio da imposição de um corte. Como o algoritmo é semelhante para todos os casos, este é apresentado uma única vez pelo Algoritmo 4. Entretanto, é destacado em

negrito (linha 10 do Algoritmo 4) apenas a parte que muda em cada situação. Os cortes relacionados à impossibilidade de atracar, calado flexível e posicionamento dinâmico são acrescentados em todos os modelos, porém nos modelos 1, 2 e 3 e 5 são obrigatórios para garantir a factibilidade da solução ótima. Ou seja, no Modelo 4 é utilizado apenas com o objetivo de melhorar o seu limitante inferior.

Nos baseamos no corte de caminhos infactíveis apresentado em Ropke et al. (2007) para aplicar os cortes relacionados à impossibilidade de atracar, calado flexível e posicionamento dinâmico. Os autores resolvem um algoritmo de separação para a seguinte restrição:

$$\sum_{(i,j)\in A(R)} x_{ij} \leq |A(R)| - 1 \quad \forall R \in \mathcal{R}. \quad (5.15)$$

sendo que R é um caminho infactível e $A(R)$ é o conjunto de arestas presentes em R.

Atracar:

Antes, porém, de detalhar o algoritmo de separação para a questão da atracação, vamos relembrar esta restrição dada no modelo de coleta e entrega e janelas de tempo e frota heterogênea para a indústria petrolífera:

$$x_{ijk} = 0 \quad \forall i,k \in A_{ik} = 1, CF_{ik} \leq 0. \quad (4.14)$$

Ou seja, se o navio k não pode atracar em i (isto é, se $A_{ik} = 1$) e se o calado de i não é flexibilizado (isto é, se $CF_{ik} \leq 0$), então este navio nunca pode atracar no ponto operacional i.

O algoritmo de separação inicia-se com todos os nós de coleta referentes aos depósitos de cada navio, ou seja, agora temos que os nós entre 1 e $|K|$ pertencem aos depósitos iniciais de cada navio em que é feita uma coleta artificial, com $|K|$ igual ao número de navios. Sendo assim, o algoritmo é executado para todo navio $i \in K$. Então, verifica-se quais os nós j em que $x_{ij} > 0$ de acordo com a solução do problema relaxado e escolhe-se um j para entrar no caminho. Neste momento, ocorre a verificação da restrição (5.15), isto é, se $\sum_{(i,j)\in A(R)} x_{ij} > |A(R)| - 1$ e também se $A_j = 1$ e se $CF_{jk} \leq 0$. Se as três hipóteses acontecerem, o algoritmo para e o corte é acrescentado. Caso contrário, continua-se o caminho até que este seja violado ou até que encontra-se o depósito final (nó en_0). O Algoritmo 4 representa o algoritmo de separação para a questão da atracação, lembrando que a parte destacada em negrito (linha 10) é a parte que se diferencia para as outras questões como calado flexível e posicionamento dinâmico.

Algoritmo 4: Algoritmo de separação para a inequação (5.15) - referente a atracação. Adaptado de (ROPKE et al., 2007)

```
para i ∈ K faça
    navio = i;
    NoAtual = i;
    R ← NoAtual;
    enquanto NoAtual > 0 faça
        para j ∈ N faça
            se x_{NoAtual,j} > 0 então
                ListaNOS ← j;
        R ← j;
        se A_{j,navio} = 1, Restrição (5.15) é violada e CF_{j,navio} ≤ 0 então
            NoAtual = −1;
            Acrescenta a restrição (5.15).
        se k = en_0 então
            NoAtual = −1;
        NoAtual = j;
```

Calado Flexível:

A restrição de calado flexível nos diz que se um navio k, que a princípio não poderia atracar em i com carga completa, possuir apenas uma porcentagem de sua carga a bordo e o calado de i estiver flexibilizado, então o navio k é autorizado a atracar em i. A restrição é dada por:

$$Q_{jk} \leq (CF_{jk}Cap_k + q_j) + \left(1 - \sum_{i \in (P \cup D \cup \{s_k\})} x_{ijk}\right) M. (4.18)$$

Este algoritmo de separação é similar com o algoritmo anterior, que analisa somente a questão de atracar. Porém, como já descrito, em algumas situações o calado do navio pode ser flexibilizado para que este possa atracar em determinados pontos operacionais. Para o algoritmo de separação, inicia-se o caminho com todos os navios $i \in K$, e analisa-se quais os nós que possuem algum fluxo com o nó i através da solução do problema relaxado. Digamos que o primeiro nó a ser verificado seja o nó j, tal que $x_{ij} > 0$, então o nó j deve ser acrescentado ao caminho R. O próximo passo é verificar se a restrição de calado flexível foi violada. Se isto ocorrer, então o algoritmo para tendo como saída o caminho R para que o corte seja acrescentado. Além disso, o algoritmo pode parar se encontrar o nó en_0 (depósito final).

Posicionamento Dinâmico:

Temos três restrições relacionadas ao posicionamento dinâmico (DP), que nos dizem se um navio pode ou não atracar em determinado ponto operacional, dependendo da combinação de fatores: se o navio possui DP e se o ponto operacional possui DP. Para cada uma das restrições, fizemos um algoritmo de separação diferente que é descrito a seguir. Antes, porém, vamos relembrar quais são estas restrições dadas no modelo de coleta e entrega e janelas de tempo para a indústria petrolífera:

$$Q_{jk} \leq (\alpha_1 Cap + q_j) + (1 - \alpha_1) Cap \left(1 - \sum_{\substack{i \in (P \cup D \cup \{s_k\}) \\ dist_{ij} > 0}} x_{ijk} \right) \quad \forall j \notin C_{DP}, k \in K_{DP}; (4.19)$$

$$Q_{jk} \leq (\alpha_2 Cap + q_j) + (1 - \alpha_2) Cap \left(1 - \sum_{\substack{i \in (P \cup D \cup \{s_k\}) \\ dist_{ij} > 0}} x_{ijk} \right) \quad \forall j \notin C_{DP}, k \notin K_{DP}; (4.20)$$

$$\sum_{i \in N} x_{ijk} = 0 \quad \forall j \in C_{DP}, k \notin K_{DP}. (4.21)$$

Para cada restrição, a linha 10 do Algoritmo 4 é verificada e se alguma destas desigualdades for violada, então um corte correspondente é acrescentado ao problema.

5.2.2 Restrição de Precedência

A restrição de precedência relembrada a seguir pertence aos Modelos 1 e 2 e é obrigatória para a garantia de factibilidade da solução ótima. Para os demais modelos é utilizada apenas para melhoria do limitante inferior.

$$\sum_{i,j \in S} x_{ij} \leq |S| - 2 \quad \forall S \in \mathcal{S}. (5.4)$$

O algoritmo de separação para gerar as restrições de precedência (5.4) é resolvido em tempo polinomial através da resolução de vários problemas de fluxo máximo (ROPKE et al., 2007). Neste tipo de problema é desejado computar o melhor custo em que podemos transportar o material da fonte até o destino final, sem violar nenhuma restrição de capacidade. O problema de fluxo máximo é um problema que engloba fluxo de redes e existem vários algoritmos eficazes para resolvê-lo. O algoritmo utilizado para determinar a restrição de precedência é o de *Ford-and-Fulkerson*. Vários autores utilizam deste algoritmo para esta finalidade (CORMEN et al., 1990; ROPKE et al., 2007).

Define-se o super nó de origem que é gerado pela junção dos nós i e en_0 e o super nó

de destino que é gerado pela junção dos nós $n+i$ e s_0. Para implementar esses super nós utilizamos de dois nós artificiais, um para o super nó de origem, e outro para o super nó de destino. A Figura 34 ilustra como isto é realizado. O nó SNO é o super nó de origem e o nó SND é o super nó de destino:

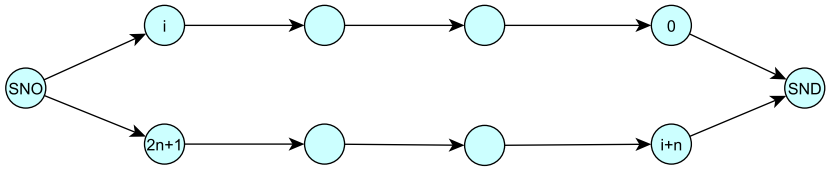

Figura 34: Exemplo de um grafo com super nós.

A capacidade do super nó é infinita, enquanto que o fluxo que passa por ele é igual à soma dos fluxos dos arcos que saem de cada nó ligado a ele. Se o valor ótimo do problema de fluxo máximo for menor que 1, a restrição de precedência é violada, ou seja, para algum nó de coleta i, o navio visita sua entrega $(n+i)$ antes de visitar i ou os nós são visitados por navios diferentes. Se o resultado for igual a 1, então não temos um corte válido. Isto acontece porque o problema de fluxo máximo nos dá o quanto de fluxo sai do nó i e chega ao nó $n+i$; sendo assim, se este resultado for igual a 1, isto quer dizer que todo o fluxo percorre o caminho desejado fazendo com que o navio visite o nó i e depois o nó $n+i$. Além disso, o resultado do problema de fluxo máximo corresponde ao corte mínimo do grafo. Desta maneira, o grafo pode ser particionado em dois para que o conjunto S seja definido pelos nós em uma das partições. Por definição, um arco é dito saturado quando o fluxo que passa por ele é igual à sua capacidade.

Na separação das restrições de precedência, o algoritmo de *Ford-and-Fulkerson* é executado para cada coleta $i \in P$, de modo a verificar se há alguma restrição de precedência violada. Se o resultado do problema de fluxo máximo for menor que 1, então existe uma restrição de precedência violada para o conjunto S dado pelos nós de um dos lados do grafo particionado e tem-se assim um corte válido. O algoritmo de separação é detalhado no Algoritmo 5:

Algoritmo 5: Algoritmo de separação para a restrição de precedência (5.4). Adaptado de (ROPKE et al., 2007)

```
para i ∈ P faça
    Defina o super nó de origem (i e en_0);
    Defina o super nó de destino (n + i e s_0);
    A capacidade máxima do arco (i,j) é dada pelo valor da solução x_ij da
    relaxação linear;
    Resolva o problema de fluxo máximo com o objetivo de sair do super nó de
    origem e chegar no super nó de destino;
    se Fluxo máximo for menor que 1 então
        Acrescente o corte (5.4) para o S encontrado.
```

5.2.3 Restrição de Capacidade

Nesta seção vamos explicar o algoritmo de separação para a restrição de capacidade (5.16). Para separar estas restrições utilizamos dois procedimentos: um baseado na meta-heurística busca tabu e outro em uma heurística construtiva que é descrita ao final desta seção. Nos modelos 1, 3, 4 e 5 tem-se a garantia de que a capacidade do navio é respeitada, então esta restrição é utilizada apenas para melhorar seu limitante inferior. Entretanto, para o Modelo 2 esta restrição é essencial para garantir a factibilidade da solução ótima, pois sem ela não temos a garantia de que a capacidade do navio seja respeitada. Antes de apresentar os algoritmos de separação, vamos relembrar a restrição de capacidade dada no Modelo 2:

$$\sum_{i,j \in S} x_{ij} \leq |S| - \max\left\{1, \lceil |q(S)| / Cap \rceil\right\}, \forall S \subseteq N \backslash \{s_0, en_0\}, |S| \geq 2 \quad (5.16)$$

5.2.3.1 Busca Tabu

A busca tabu é uma meta-heurística muito utilizada em problemas de otimização e consiste em buscar soluções de melhor qualidade numa vizinhança, mesmo que nesta vizinhança tenha sido encontrada uma solução de pior qualidade. Acredita-se que, após algumas iterações, seja possível encontrar uma solução melhor.

Este algoritmo de separação é inspirado em Augerat et al. (1998), em que foi descrito uma busca tabu para restrições do CVRP (problema de roteirização de veículos capacitado). Algumas adaptações são feitas para que este algoritmo de separação seja válido também para o problema de coleta e entrega conforme apresentado em Ropke et al. (2007). Cada restrição de capacidade está associada a um número de navios necessários

para atender a todos os nós de um dado conjunto S, com $S \subset N \setminus \{s_0, en_0\}$. Um parâmetro p varia a cada iteração e indica o número de navios, ou seja, se $p = 1$, então possui 1 navio, e assim sucessivamente. Para cada valor de p procura-se encontrar conjuntos de nós que requerem p navios.

O algoritmo se baseia em duas fases: a primeira de construção do conjunto S denominada *expansion*, e a segunda de mudança dos elementos de S a fim de melhorar a violação do corte determinado por S, denominada *interchange*. A primeira fase inicia-se com um elemento de $N \setminus \{s_0, en_0\}$. Este procedimento é realizado para metade do número de nós, ou seja, para $n/2$ nós. Então, é sorteado aleatoriamente qual elemento inicia o procedimento. Sendo assim, dado um elemento sorteado aleatoriamente, digamos i, então $S \leftarrow i$ e inicia-se a primeira fase. Através de conjuntos formados por elementos que podem ser removidos de S e outro conjunto dos elementos que podem ser adicionados a S, o algoritmo define na segunda fase, qual elemento deve sair de S ou qual elemento deve entrar em S. Sendo assim, é necessário ter uma lista tabu a fim de evitar ciclos. Então, se um elemento sai de S ou entra em S, este elemento não pode ser modificado novamente durante um certo número de iterações denotado por *tll*.

A primeira fase é aplicada até que a restrição de capacidade seja violada, ou seja, até que $\sum_{i,j \in S} x_{ij} > |S| - \max\{1, \lceil |q(S)| / Cap \rceil\}$. A segunda fase é aplicada durante um certo número de iterações dado por um parâmetro denominado *tope*. O algoritmo todo (fase 1 + fase 2) é executado *ntimes* vezes.

Em Ropke et al. (2007) os autores propõem que inicialmente seja sorteado um subconjunto $S \subset P$ e depois o algoritmo seja executado novamente sorteando-se um subconjunto de $S \subset D$. Optamos inicialmente por sortear dois elementos de cada conjunto. O algoritmo então roda $n/4$ vezes para os nós de coleta sorteados inicialmente e $n/4$ vezes para os nós de entrega. Cabe ressaltar que esta distinção entre os nós de coleta e entrega somente é feita nesta parte do algoritmo; durante sua execução podem entrar ou sair de S quaisquer nós em $N \setminus \{s_0, en_0\}$.

Selecionado aleatoriamente um subconjunto de nós, o algoritmo é executado para $p = 1$ navio, $p = 2$ navios, até $p = k - 1$ navios, em que k é a quantidade total de navios do exemplar em questão. Quando p é incrementado resolve-se novamente a primeira e depois a segunda fase.

Dado um conjunto S de $N \setminus \{s_0, en_0\}$, denota-se por $N^+(S)$ o conjunto dos nós em $N \setminus \{s_0, en_0\} \setminus S$ adjacentes à rede $G(x)$ a pelo menos um nó em S, e por $N^-(S)$ o conjunto

dos nós em S adjacentes a pelo menos um nó em $N \setminus \{s_0, en_0\} \setminus S$. A Figura 35 exemplifica um conjunto S em uma das fases do algoritmo da busca tabu. Neste exemplo, $S = \{4, 5\}$ e $N \setminus \{s_0, en_0\} \setminus S = \{1, 2, 3, 6\}$. Além disso, $N^+(S) = \{3, 6\}$ e $N^-(S) = \{4, 5\}$.

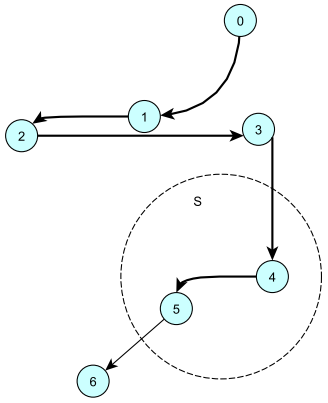

Figura 35: Exemplo do conjunto S em uma das fases do algoritmo.

A definição original em Augerat et al. (1998) para os seguintes termos é $smax = Cap(p + ulimit) - q(S)$ e $smin = q(S) - Cap(p + llimit)$, em que $ulimit$ e $llimit$ são dois parâmetros que estão entre 0 e 1. Sendo assim, o conjunto de candidatos a serem adicionados a S é dado por $C^+(S) = \{v \in N^+(S) : q_v \leq smax\}$ e o conjunto de candidatos a serem removidos de S é dado por $C^-(S) = \{v \in N^-(S) : q_v \leq smin\}$, sendo que q_v é a demanda do nó v. Se na definição de $C^+(S)$ não tivesse a tolerância $ulimit$, os nós acrescentados ao conjunto S não iriam violar a capacidade do navio, implicando em não obter um corte. Como temos esta tolerância, podemos adicionar um nó que, em algum momento, a capacidade do navio será violada. De forma análoga, temos a tolerância $llimit$ em $C^-(S)$ para que, ao tirar um nó do conjunto S, ainda tenhamos um corte válido. Para adaptar o algoritmo de Augerat et al. (1998) para o problema de coleta e entrega, as definições de $C^+(S)$ e de $C^-(S)$ são modificadas da seguinte forma:

$$C^+(S) = \left\{v \in N^+(S) : |q(S) + q_v| \leq Cap(p + ulimit)\right\}$$

$$C^-(S) = \left\{v \in N^-(S) : |q(S) - q_v| \geq Cap(p - llimit)\right\}$$

Estas modificações são necessárias para que demandas positivas (coletas) e negativas (entregas) sejam consideradas. Na fase *expansion* o melhor candidato a ser adicionado a S é aquele em que $x((S : \{v\}))$ é máximo, ou seja, dos nós que fazem fronteira com o conjunto S (nós que estão fora de S, mas que possuem algum arco ligado a S), escolhe-se

aquele em que x_{ij} é maior, sendo que i é o nó que está em S e j é o nó v. Entretanto, a fim de obter diferentes resultados ao executar o algoritmo algumas vezes, este passo é feito de maneira aleatória: constrói-se um conjunto de bons candidatos e seleciona-se um nó aleatoriamente dentro deste conjunto. Então, calcula-se o arco máximo para que este entre no conjunto S e minimize $x(\delta(S))$. Um parâmetro denominado *per*, com $0 \leq per \leq 1$, é utilizado para definir este conjunto de bons candidatos a entrar em S.

A Figura 36 ilustra uma das iterações do algoritmo. Neste ponto, temos que $S = \{1, 2\}$. Os candidatos a entrarem no conjunto S são os nós 3 e 5, se ambos pertencem a $C^+(S)$ (isto depende do cálculo de $N^+(S)$). Vamos supor que os dois nós sejam bons candidatos a entrar no conjunto S, então calcula-se $M = max\{x((S:v)) : v \in C^+(S)\}$ e aleatoriamente seleciona-se um nó $v \in C^+(S)$ dos que satisfazem $M - per \leq x((S:v)) \leq M$. Acrescenta-se o nó sorteado a S e a restrição de capacidade é verificada. Se esta não for violada, calculamos novamente $N^+(S)$ e $C^+(S)$. Se $C^+(S)$ for vazio, então a primeira fase do algoritmo se encerrou e inicia-se a segunda fase (*interchange*); caso contrário, repete-se o procedimento citado acima, até que a restrição seja violada.

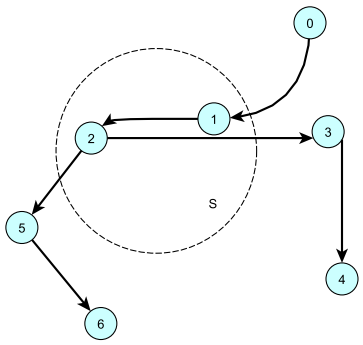

Figura 36: Exemplo do conjunto S em uma das fases do algoritmo.

Na fase *interchange* calcula-se $N^+(S)$, $C^+(S)$, $N^-(S)$ e $C^-(S)$. Remove-se de $C^-(S)$ todos os nós que são adicionados em S nas últimas *tll* iterações. Isto é feito para evitar ciclos. Do mesmo modo, remove-se de $C^+(S)$ todos os nós que são removidos de S nas últimas *tll* iterações, para que um nó que esteja declarado como tabu não seja escolhido. O próximo passo do algoritmo é calcular:

$$max\left\{\left\{x((S:j)), j \in C^+(S)\right\}, \left\{x((N \setminus \{s_0, en_0\} \setminus S:j)), j \in C^-(S)\right\}\right\}.$$

Se o nó v pertencer a $C^+(S)$, então adiciona-se v em S; se v pertencer a $C^-(S)$, então

deve-se remover v de S. O passo seguinte é verificar a restrição de capacidade e analisar se esta foi violada.

O Algoritmo 6 representa o algoritmo completo da busca tabu para separar restrições de capacidade:

Algoritmo 6: Algoritmo de Busca Tabu. Adaptado de (AUGERAT et al., 1998)

```
1  Selecione S ← i;
2  Selecione p = 1;
3  Fase expansion:
4  enquanto p ≤ k − 1 faça
5      enquanto Restrição de capacidade não é violada faça
6          E1: Computar C+(S). Se C+(S) é vazio vá para a linha 10;
7          E2: Computar M = max{x((S : v)) : v ∈ C+(S)} e aleatoriamente
           selecionar um nó v ∈ C+(S) dos que satisfazem M − per ≤ x((S : v)) ≤ M;
8          E3: Adicionar v a S e checar a restrição de capacidade (5.16).
9      Fase interchange:
10     I0: iter = 1;
11     enquanto iter < tope faça
12         enquanto Restrição de capacidade não é violada faça
13             I1: Computar C+(S) e C−(S). Remover de C−(S) (C+(S)) os nós
               que são adicionados (removidos) de S em qualquer das últimas tll
               iterações. Se C+(S) ∪ C−(S) é vazio vá para a linha 17;
14             I2: Seja v o nó em que
               max{{x((S : j)), j ∈ C+(S)}, {x((N \ {s0,en0} \ S : j)), j ∈ C−(S)}};
15             I3: Dependendo se v ∈ C+(S) ou v ∈ C−(S), adicione ou remova de S
               o nó v e checar a restrição de capacidade (5.16);
16             iter = iter + 1;
17     I4: p = p + 1.
```

Os parâmetros utilizados são definidos da seguinte maneira (AUGERAT et al., 1998):

- *ntimes*: entre 1 e 6;
- *ulimit*: $0{,}3 \leq ulimit \leq 0{,}45$;
- *llimit*: $0{,}1 \leq ulimit \leq 0{,}25$;
- *tll*: $5 \leq tll \leq 15$;
- Se $ntimes \geq 2$, uma boa escolha para *per* é $[0{,}2, 0{,}6]$, caso contrário, $per = 0$;
- *tope*: entre 5 e 60.

Pode ser utilizado algum critério para selecionar quais cortes serão inseridos no problema como, por exemplo, acrescentar somente os cortes que sejam mais violados. Cada vez que o algoritmo encontrar um corte que seja violado, este é acrescentado ao problema relaxado junto com as demais restrições.

5.2.3.2 Heurística Construtiva

A seguir, descreve-se a segunda heurística utilizada como algoritmo de separação para a restrição de capacidade. Esta heurística inicia-se sorteando um nó $i \in P \cup D$ e gradualmente adiciona-se nós a S considerando os nós que possuem algum fluxo de acordo com a solução do problema relaxado. Dados dois índices j e k tais que $x_{ij} > 0$ e $x_{ik} > 0$, então escolhemos o nó j ou o nó k para entrar em S. Esta escolha é feita de maneira aleatória sendo que, o nó que possuir maior fluxo, tem maiores chances de ser o escolhido. Este procedimento é realizado escolhendo vários nós iniciais e, enquanto a restrição de capacidade não for violada, outros nós podem ser adicionados ao conjunto S. Em algum momento, obtem-se um conjunto S tal que a restrição é violada, ou seja, $\sum_{i,j \in S} x_{ij} > |S| - \max\{1, \lceil |q(S)| / Cap \rceil\}$. O Algoritmo 7 mostra o algoritmo de separação desta heurística construtiva para a restrição de capacidade:

Algoritmo 7: Heurística construtiva para a restrição (5.16)

```
1  Resolva relaxação linear do modelo;
2  para i ∈ N\{s_0,en_0} faça
3      S ← i;
4      enquanto Existir j tal que x_ij > 0 ou x_ji > 0 na solução do problema relaxado
       faça
5          Acrescente j à ListaNOSCap;
6      se Não existir j então
7          Pare.
8      Escolha j em ListaNOSCap;
9      S ← j;
10     se Restrição de capacidade (5.16) foi violada então
11         Acrescente o corte correspondente e pare.
12     senão
13         Volte para a linha 4.
```

Todos os nós j encontrados tais que $x_{ij} > 0$ ou $x_{ji} > 0$ são acrescidos em uma lista de nós denominada *ListaNOSCap*, a quantidade de vezes proporcional ao valor de x_{ij} ou x_{ji} na solução do problema relaxado. Isto é feito para que, ao escolher algum nó em

ListaNOSCap, um nó j em que o valor de x_{ij} ou x_{ji} do problema relaxado seja mais próximo de 1, tenha probabilidade maior de ser escolhido. Sendo assim, temos que:

- Se $0 < x_{ij} \leq 0{,}2$ acrescenta-se este j uma vez na lista;
- Se $0{,}2 < x_{ij} \leq 0{,}4$ acrescenta-se este j duas vezes na lista;
- Se $0{,}4 < x_{ij} \leq 0{,}6$ acrescenta-se este j três vezes na lista;
- Se $0{,}6 < x_{ij} \leq 0{,}8$ acrescenta-se este j quatro vezes na lista;
- Se $0{,}8 < x_{ij} \leq 1$ acrescenta-se este j cinco vezes na lista.

Observe que se a solução do problema relaxado for inteira, ou seja, todas as variáveis x_{ij} são iguais a 1, então este algoritmo se torna exato, uma vez que percorre toda solução e faz a verificação se a capacidade foi ou não respeitada.

5.2.4 Eliminação de sub-rota

Antes de descrever o algoritmo de separação para as duas inequações de eliminação de sub-rota, vamos recordar algumas definições apresentadas na Seção 3.4. Dado um conjunto $S \subseteq P \cup D$ e seu complementar $N \backslash S$, seja $\pi(S) = \{i \in P | n + i \in S\}$ o conjunto dos predecessores de S e $\sigma(S) = \{n + i \in D | i \in S\}$ o conjunto dos sucessores de S. Desta maneira, as seguintes inequações são válidas:

$$x(S) + \sum_{i \in N \backslash S \cap \sigma(S)} \sum_{j \in S} x_{ij} + \sum_{i \in N \backslash S \backslash \sigma(S)} \sum_{j \in S \cap \sigma(S)} x_{ij} \leq |S| - 1; \quad (3.17)$$

$$x(S) + \sum_{i \in S} \sum_{j \in N \backslash S \cap \pi(S)} x_{ij} + \sum_{i \in S \cap \pi(S)} \sum_{j \in N \backslash S \backslash \pi(S)} x_{ij} \leq |S| - 1. \quad (3.18)$$

Esta restrição tem o objetivo de melhorar o limitante inferior de todos os modelos apresentados, não sendo obrigatória em nenhum deles. O algoritmo de separação descrito na sequência não é igual ao da restrição clássica de eliminação de sub-rota (3.16), em que resolve-se vários problemas de fluxo máximo (CORMEN et al., 1990). Aqui o algoritmo de separação para as inequações (3.17) e (3.18) é dado resolvendo-se novamente uma busca tabu similar à resolvida para a restrição de capacidade (Seção 5.2.3.1). Esta busca tabu possui também duas fases: uma de construção do conjunto S e outra de modificação deste conjunto, de modo a obter um corte com maior violação. O algoritmo é semelhante

ao descrito para a inequação de capacidade, tendo apenas algumas adaptações para esta inequação que são descritas a seguir.

A primeira modificação é que o algoritmo inicia-se com apenas um elemento no conjunto S, ou seja, sorteamos um nó de $i \in P \cup D$ para iniciar o procedimento, então $S \leftarrow i$. Lembramos que os depósitos inicial e final não entram no conjunto S. A definição de $N^+(S)$ e $N^-(S)$ continua a mesma: dado um subconjunto S de $N \setminus \{s_0, en_0\}$, denotamos por $N^+(S)$ o conjunto dos nós em $N \setminus \{s_0, en_0\} \setminus S$ adjacentes à rede $G(x)$ a pelo menos um nó em S, e por $N^-(S)$ o conjunto dos nós em S adjacentes a pelo menos um nó em $N \setminus \{s_0, en_0\} \setminus S$.

Outra modificação é a definição de $C^+(S)$ e $C^-(S)$. Para adaptar estas duas definições para a inequação de eliminação de sub-rota, vamos primeiramente analisar como $C^+(S)$ é obtido para a inequação de capacidade (5.16). Antes, tínhamos que $q(S) \leq Cap$, o que implicava em

$$C^+(S) = \left\{ v \in N^+(S) : |q(S) + q_v| \leq Cap\,(p + ulimit) \right\}$$

Agora, temos que $x(S) \leq |S| - 1$, o que implica na definição de $C^+(S)$:

$$C^+(S) = \left\{ v \in N^+(S) : x(S) + x_{uv} \leq (|S| - 1)\,(p + tol)\,, u \in S \right\}$$

O mesmo ocorre para o $C^-(S)$. Antes, tínhamos que:

$$C^-(S) = \left\{ v \in N^-(S) : |q(S) - q_v| \geq Cap\,(p - llimit) \right\}$$

e agora,

$$C^-(S) = \left\{ v \in N^-(S) : x(S) - x_{uv} \geq (|S| - 1)\,(p - tol)\,, u \notin S \right\}$$

sendo que *tol* é uma tolerância e possui valor entre 0 e 1. Note que, na primeira iteração do algoritmo, temos $|S| = 1$ o que resulta em um $C^+(S)$ vazio. Então, sorteamos qualquer nó que esteja em $N^+(S)$ na primeira iteração. A partir da segunda, o cálculo de $C^+(S)$ continua o mesmo. Além disso, a restrição de eliminação de sub-rota é verificada, ao invés da restrição de capacidade. Apenas estas características diferem da busca tabu já descrita anteriormente.

5.2.5 *Restrições de Ordem Generalizadas*

Esta inequação é verificada para todos os métodos, entretanto somente é utilizada para melhorar os limitantes inferiores. Primeiramente, vamos relembrar algumas definições do Capítulo 3: seja U_1,...,$U_m \subset P$ tal que $s_0, en_0 \notin U_l$ e i_l,$n + i_{l+1} \in U_l$, para $l = 1$,...,m em que $i_{m+1} = i_1$. O algoritmo de separação é aplicado somente para as duas inequações mais fortes dadas a seguir:

$$\sum_{l=1}^{s} x\left(U_l\right) + \sum_{l=2}^{s-1} x_{i_1,i_l} + \sum_{l=3}^{s} x_{i_1,n+i_l} \leq \sum_{l=1}^{s} |U_l| - s - 1; (3.22)$$

$$\sum_{l=1}^{s} x\left(U_l\right) + \sum_{l=2}^{s-2} x_{n+i_1,i_l} + \sum_{l=2}^{s-1} x_{n+i_1,n+i_l} \leq \sum_{l=1}^{s} |U_l| - s - 1. (3.23)$$

Além disso, o algoritmo é aplicado para um caso especial das restrições, isto é, quando $m = 3$ e $|U_1| = |U_2| = |U_3| = 2$. Para a inequação (3.22), tem-se que o algoritmo é executado para cada nó $i \in P$. Em seguida, o objetivo é encontrar um nó j tal que maximize $x_{i,n+j}$, $+ x_{n+j,i} + x_{i,j}$. Depois, tem-se que encontrar um nó k tal que o lado esquerdo da inequação (3.22) seja maximizado. Os valores das variáveis x são dados através da solução do problema relaxado. O Algoritmo 8 representa o algoritmo de separação para a inequação (3.22).

Para a inequação (3.23) o algoritmo de separação é similar: este é executado para cada nó de coleta $i \in P$. Desta maneira, o objetivo é encontrar um nó j tal que maximize $x_{i,n+j}$, $+ x_{n+j,i} + x_{n+i,n+j}$. Depois, tem-se que encontrar um nó k tal que o lado esquerdo da inequação (3.23) seja maximizado. O Algoritmo 9 representa o algoritmo de separação para a inequação (3.23).

5.2.6 Caminhos Infactíveis

Estas inequações juntamente com o algoritmo de separação já foram descritos na seção referente à verificação da impossibilidade de atracar, calado flexível e posicionamento dinâmico (Seção 5.2.1). Entretanto, optamos por separar as contribuições do estudo de caso com as inequações e algoritmos de separação já presentes na literatura e, portanto, segue agora a parte referente às janelas de tempo das inequações de caminhos infactíveis.

Esta inequação pertence ao Modelo 2 que foi descrito na Seção 5.1.2. Para este modelo, esta inequação é essencial para que se tenha a garantia da factibilidade da solução ótima

Algoritmo 8: Algoritmo de separação para a inequação (3.22). Adaptado de (CORDEAU, 2006)

```
1  para i ∈ P faça
2      i_1 = i;
3      i_2 = 0;
4      i_3 = 0;
5      M = 0;
6      M2 = 0
7      para j ∈ P faça
8          se M < x_{i,n+j}, + x_{n+j,i} + x_{i,j} então
9              M = x_{i,n+j}, + x_{n+j,i} + x_{i,j};
10             i_2 = j;
11     se M ≠ 0 então
12         j = i_2;
13         para k ∈ P faça
14             se M2 < x_{i,n+j}, + x_{n+j,i} + x_{j,n+k} + x_{n+k,j}, + x_{k,n+i} + x_{n+i,k} + x_{i,j} + x_{i,n+k}
               então
15                 M2 = x_{i,n+j}, + x_{n+j,i} + x_{j,n+k} + x_{n+k,j}, + x_{k,n+i} + x_{n+i,k} + x_{i,j} + x_{i,n+k};
16                 i_3 = k;
17     se M ≠ 0 e M2 ≠ 0 então
18         Acrescente o corte correspondente a restrição (3.22).
```

Algoritmo 9: Algoritmo de separação para a inequação (3.23). Adaptado de (CORDEAU, 2006)

```
1  para i ∈ P faça
2      i_1 = i;
3      i_2 = 0;
4      i_3 = 0;
5      M = 0;
6      M2 = 0
7      para j ∈ P faça
8          se M < x_{i,n+j}, + x_{n+j,i} + x_{n+i,n+j} então
9              M = x_{i,n+j}, + x_{n+j,i} + x_{n+i,n+j};
10             i_2 = j;
11     se M ≠ 0 então
12         j = i_2;
13         para k ∈ P faça
14             se M2 < x_{i,n+j}, + x_{n+j,i} + x_{j,n+k} + x_{n+k,j}, + x_{k,n+i} + x_{n+i,k} + x_{n+i,n+j}
               então
15                 M2 = x_{i,n+j}, + x_{n+j,i} + x_{j,n+k} + x_{n+k,j}, + x_{k,n+i} + x_{n+i,k} + x_{n+i,n+j};
16                 i_3 = k;
17     se M ≠ 0 e M2 ≠ 0 então
18         Acrescente o corte correspondente a restrição (3.23).
```

em relação às janelas de tempo. Para os demais modelos, este corte é utilizado somente para melhorar seu limitante inferior. Apenas para relembrar, segue a inequação:

$$\sum_{(i,j)\in A(R)} x_{ij} \leq |A(R)| - 1 \quad \forall R \in \mathcal{R}. (5.15)$$

A próxima inequação foi dada no Capítulo 3 e refere-se a caminhos em que unimos o par de nós i e $n+i$ em R, ou seja, $R = (i,k_1,k_2,...,k_r,n+i)$:

$$x_{i,k_1} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + x_{k_r,n+i} \leq |A(R)| - 2. \quad (3.26)$$

Para a inequação (5.15) é utilizado um procedimento enumerativo similar ao descrito em Ascheuer et al. (2001) e similar ao descrito para as inequações que verificam a impossibilidade de atracar, calado flexível e posicionamento dinâmico, porém aqui as janelas de tempo é que são verificadas. Para cada nó $i \in P \cup D$, temos que i é o nó de origem deste caminho R. Temos que verificar quais nós possuem algum fluxo com i de acordo com a solução do problema relaxado, ou seja, se existe algum nó j, tal que $x_{ij} > 0$. Então, o nó j é acrescentado ao caminho R. A partir do nó j, verifica-se quais nós possuem algum fluxo com j em relação à solução do problema relaxado, ou seja, se existe algum nó k tal que $x_{jk} > 0$ e acrescenta este nó k em R e, assim, sucessivamente, formando uma árvore de caminhos. O procedimento para quando encontra-se o nó en_0 (depósito final) ou quando a restrição é violada, ou seja, alguma janela de tempo é violada. Além disso, verificamos a cada iteração, se a restrição de caminhos infactíveis é violada, ou seja, se o lado esquerdo da inequação (5.15) é menor ou igual a $|A(R)| - 1$. Se for estritamente maior, é porque a inequação é violada. O algoritmo é similar ao Algoritmo 4 apresentado anteriormente, mudando apenas a linha 10 referente à verificação das janelas de tempo.

O algoritmo de separação para a inequação (3.26) é adaptado de Cordeau (2006). Para esta inequação, o algoritmo é executado apenas para os nós de coleta ($i \in P$). Para cada i, construímos um caminho considerando apenas os nós com maior fluxo de acordo com a solução do problema relaxado. Então, apenas os nós com maior fluxo é que serão acrescentados em R. O algoritmo para quando forma-se um ciclo (nó repetido), quando encontra-se o depósito final (en_0) ou quando encontra-se o nó $n+i$ (entrega respectiva do nó i). Quando o nó $n+i$ é encontrado, então a restrição de janela de tempo é verificada, ou seja, se $B_j \geq B_i + d_i + t_{ij} - BigM(1 - x_{ij}), \forall i \in N, j \in N$. Se esta desigualdade é violada, então acrescenta-se o corte correspondente. O Algoritmo 10 representa o algoritmo de separação para a inequação 3.26.

Algoritmo 10: Algoritmo de separação para a inequação (3.26). Adaptado de (CORDEAU, 2006)

```
1  para i ∈ P faça
2      NoAtual = i;
3      R[auxr] = NoAtual;
4      auxr++;
5      enquanto NoAtual > 0 faça
6          k = 0;
7          M = 0;
8          para j ∈ N faça
9              se x_{NoAtual,j} > 0  e  M < x_{NoAtual,j} então
10                 M = x_{NoAtual,j};
11                 k = j;
12         se M = 0 então
13             NoAtual = −1;
14         senão
15             se O nó é repetido então
16                 NoAtual = −1;
17             senão
18                 R[auxr] = NoAtual;
19                 auxr++;
20                 NoAtual = k;
21             se k = n + i então
22                 NoAtual = −1;
23                 Verifica se violou a restrição de janela de tempo;
24                 se Violou a inequação então
25                     Insira o corte (3.26);
26             se k = en_0 então
27                 NoAtual = −1;
```

5.2.7 Restrições de Alcance

Estas inequações são utilizadas em todos os modelos para melhorar seus limitantes inferiores e são relembradas a seguir juntamente com algumas definições dadas no Capítulo 3. Seja $\delta(S) = \delta^+(S) \cup \delta^-(S)$, em que $\delta^+(S) = \{(i,j) \in A | i \in S, j \notin S\}$ e $\delta^-(S) = \{(i,j) \in A | i \notin S, j \in S\}$. Ainda, para cada nó $i \in N$, seja $A_i^- \subset A$ o conjunto mínimo de arcos, tal que, qualquer caminho factível que inicie em s_0 e termine em i, utiliza somente arcos de A_i^-. Seja A_i^+ o conjunto mínimo de arcos, tal que, qualquer caminho factível que se inicie em i e termine em en_0, utiliza somente arcos de A_i^+. Considere o conjunto de nós conflitantes T tais que, cada nó em T, deve ser visitado por um navio diferente. Sendo assim, para qualquer conjunto de nós conflitantes T seja $A_T^- = \cup_{i \in T} A_i^-$ e $A_T^+ = \cup_{i \in T} A_i^+$. Para cada conjunto $S \subseteq P \cup D$ e cada conjunto do tipo conflitante $T \subseteq S$, temos que:

$$x\left(\delta^-(S) \cap A_T^-\right) \geq |T|, \quad (3.29)$$

$$x\left(\delta^+(S) \cap A_T^+\right) \geq |T|. \quad (3.30)$$

A ideia básica do algoritmo de separação para a restrição de alcance é identificar primeiramente todos os arcos pertencentes aos conjuntos A_i^- e A_i^+ analisando a capacidade, precedência, janelas de tempo e também as questões relacionadas a impossibilidade de atracar, calado flexível e posicionamento dinâmico. Depois, identificar todos os conjuntos T de nós conflitantes (como são muitos os conjuntos T, identificar primeiramente somente os conjuntos T com cardinalidade 2). Então, resolve-se um problema de fluxo máximo entre o nó s_0 e o conjunto T, considerando somente os arcos em A_T^-. Se a capacidade do corte mínimo for menor que $|T|$, então um corte válido é encontrado. O mesmo é feito para A_T^+ em que resolve-se o problema de fluxo máximo entre o conjunto T e o depósito final en_0. Se o resultado do problema do fluxo máximo for menor que $|T|$, então um corte é encontrado.

O problema está em identificar todos os conjuntos A_k^- e A_k^+, pois são muitas as possibilidades. Inicialmente, poderíamos colocar quaisquer arcos nestes conjuntos, entretanto não teríamos conjuntos de nós conflitantes. Sendo assim, temos que analisar se existe algum caminho factível de cada arco (i,j) para um nó k, se existir, então este arco é inserido no conjunto A_k^-, caso contrário, este arco não é inserido no conjunto.

Para cada nó $k \in N \setminus \{s_0, en_0\}$, devemos analisar cada arco (i,j) e conferir se este arco pertence a algum caminho factível do tipo $(s_0,...,i,j,...,k)$. A verificação deve ser feita em relação à capacidade, janela de tempo, precedência, impossibilidade de atracar, calado

flexível e posicionamento dinâmico. Se este arco pertencer a algum caminho factível, então ele é acrescentado em A_k^-, caso contrário não é acrescentado ao conjunto. Para o arco não entrar no conjunto A, este tem que violar todas as opções propostas, ou seja, não pode existir nenhum caminho que seja factível que inicie sua rota em s_0, passe por (i,j) e viaje até o nó k. As possibilidades em relação à janela de tempo devem ser analisadas juntamente com a possibilidade de violação da capacidade. A seguir apresenta-se um exemplo do que deve ser verificado para cada i,j,k; as outras opções a serem verificadas encontram-se no Apêndice A.

Se i,j,k são nós de coleta (PPP):

- Os três nós precisam ser diferentes;
  - $(s_0,...,i,j,...,k)$ e capacidade $q_i + q_j + q_k \leq Cap$;
  - $(s_0,...,i,j,...,i+n,...,j+n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_k \leq Cap$;
  - $(s_0,...,i,j,...,j+n,...,i+n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_k \leq Cap$;
  - $(s_0,...,i,j,...,i+n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_j + q_k \leq Cap$;
  - $(s_0,...,i,j,...,j+n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_i + q_k \leq Cap$.

Depois desta fase de identificação de todos os conjuntos A, o próximo passo é identificar os conjuntos T de nós conflitantes. Dizemos que um nó j é um possível predecessor de um nó i se, e somente se, existe um $k \in N \setminus \{s_0, en_0\}$ tal que $(j,k) \in A_i^-$. Neste contexto, dois nós i,j são conflitantes se, e somente se, i não é um possível predecessor de j, e j não é um possível predecessor de i. Note que o arco sempre "sai" de j e não um arco que "chega" em j, ou seja, um arco do tipo $(j,\_)$. Um conjunto T é um conjunto de nós conflitantes se, e somente se, cada par $i,j \in T$ é conflitante. Sendo assim, verificamos se dois nós i,j são conflitantes se não existe um possível k em cada um dos conjuntos A.

Após encontrar todos os possíveis conjuntos T formados por dois elementos, então um problema de fluxo máximo é resolvido para cada conjunto T, entre o nó inicial s_0 e o conjunto T. Os arcos que são inseridos no grafo para a resolução do problema de fluxo máximo são os arcos presentes em A_T^- e as capacidades são definidas pelo valor da solução do problema relaxado na iteração específica. Se o resultado do problema de fluxo máximo for menor que $|T|$, então temos um corte válido. O resultado do problema de fluxo máximo corresponde ao corte mínimo do grafo e como já dito, o grafo pode ser particionado em dois para que o conjunto S seja definido pelos nós em uma das partições.

Para determinar os conjuntos $A_k^+, \forall k \in P \cup D$, temos que verificar todos os arcos (i,j) que pertencem a caminhos factíveis do tipo (k,...,i,j,...,en_0). O mesmo que foi realizado para A_k^- é feito para os conjuntos A_k^+ (a cada caminho é feita a verificação da precedência, janela de tempo, capacidade, atracar, calado flexível e posicionamento dinâmico). Novamente vamos exemplificar apenas uma situação, as outras encontram-se no Apêndice A.

Se os três nós são nós de coleta (PPP):

- Os três nós são diferentes ($i \neq j \neq k$):
  - (k,...,$k+n$,...i,j,...,$j+n$,...,$i+n$,...,en_0) e capacidade $q_k \leq Cap$ e q_i+$q_j \leq Cap$;
  - (k,...,$k+n$,...i,j,...,$i+n$,...,$j+n$,...,en_0) e capacidade $q_k \leq Cap$ e q_i+$q_j \leq Cap$;
  - (k,...i,j,...,$k+n$,...,$i+n$,...,$j+n$,...,en_0) e capacidade $q_i + q_j + q_k \leq Cap$;
  - (k,...i,j,...,$k+n$,...,$j+n$,...,$i+n$,...,en_0) e capacidade $q_i + q_j + q_k \leq Cap$;
  - (k,...i,j,...,$j+n$,...,$k+n$,...,$i+n$,...,en_0) e capacidade $q_i + q_j + q_k \leq Cap$;
  - (k,...i,j,...,$j+n$,...,$i+n$,...,$k+n$,...,en_0) e capacidade $q_i + q_j + q_k \leq Cap$;
  - (k,...i,j,...,$i+n$,...,$j+n$,...,$k+n$,...,en_0) e capacidade $q_i + q_j + q_k \leq Cap$;
  - (k,...i,j,...,$i+n$,...,$k+n$,...,$j+n$,...,en_0) e capacidade $q_i + q_j + q_k \leq Cap$.

Depois desta fase de identificação de todos os conjuntos A, o próximo passo é identificar os conjuntos T de nós conflitantes. Dizemos que um nó j é um possível predecessor de um nó i se, e somente se, existe um $k \in N \setminus \{s_0, en_0\}$ tal que $(k,j) \in A_i^-$. Neste contexto, dois nós i,j são conflitantes se, e somente se, i não é um possível predecessor de j, e j não é um possível predecessor de i. Note que o arco sempre "chega" em j e não um arco que "sai" de j, ao contrário do que acontece com os conjuntos A^-, ou seja, um arco do tipo (_,j). Um conjunto T é um conjunto de nós conflitantes se, e somente se, cada par i,$j \in T$ é conflitante.

Após encontrar todos os possíveis conjuntos T formados por dois elementos, então um problema de fluxo máximo é resolvido para cada conjunto, entre o conjunto T e o nó final en_0. Os arcos que são inseridos no grafo para a resolução do problema de fluxo máximo são os arcos presentes em A_T^+ e as capacidades são definidas pelo valor da solução ótima do problema relaxado. Se o resultado do problema de fluxo máximo for menor que $|T|$, então temos um corte válido.

5.3 Pré-Processamento

Em Cordeau (2006) e Dumas et al. (1991), os autores mostram uma série de eliminação de arcos que podem ser feitas no pré-processamento dos modelos apresentados. Alguns cortes também podem ser colocados diretamente no modelo para que sejam verificados em cada nó da árvore *branch-and-bound* antes de iniciar os algoritmos de separação. Isto é realizado para diminuir o tempo computacional nos algoritmos e consequentemente, diminuir o tempo de processamento em cada nó. Primeiramente, vamos discutir os arcos que podem ser pré-fixados, depois os cortes que podem ser colocados nos modelos de 2-índices (os cortes para os modelos de 3-índices já foram apresentados na Seção 4.4) e, por último, o pré-processamento específico para o problema do estudo de caso.

Arcos:

1. Arcos $(s_0, n+i)$, (i, en_0), $(n+i, i)$, (en_0, s_0), (en_0, i), $(en_0, n+i)$, (i, i), (i, s_0), $(n+i, s_0)$ e $(n+i, n+i)$ são infactíveis para $i \in P$;

2. Arco (i, j) com i,$j \in N$ é infactível se $e_i + d_i + t_{ij} > l_j$;

3. Arcos (i, j) e $(j, n+i)$ com $i \in P$, $j \in N$ são ambos infactíveis se $t_{ij} + d_j + t_{j,n+i} > l_{en_j}$;

4. Capacidade: Se $q_i + q_j > Cap \quad \forall i,j \in P$ com $i \neq j$ então os arcos (i, j), (j, i), $(i, n+j)$, $(j, n+i)$, $(n+i, n+j)$ e $(n+j, n+i)$ podem ser eliminados;

5. Janela de tempo: Se $e_i + d_i + t_{ij} > l_j \quad \forall i,j \in P \cup D$, então o arco (i, j) pode ser eliminado.

Combinando as janelas de tempo com os pares de coleta/entrega, pode-se encontrar outras regras de eliminação de arcos:

1. Arco $(i, n+j)$ é infactível se o caminho $\{j, i, n+j, n+i\}$ for infactível;

2. Arco $(n+i, j)$ é infactível se o caminho $\{i, n+i, j, n+j\}$ for infactível;

3. Arco (i, j) é infactível se os caminhos $\{i, j, n+i, n+j\}$ e $\{i, j, n+j, n+i\}$ forem infactíveis;

4. Arco $(n+i, n+j)$ é infactível se os caminhos $\{i, j, n+i, n+j\}$ e $\{j, i, n+i, n+j\}$ forem infactíveis.

Ao se levar em conta as relações de precedência, juntamente com as janelas de tempo, pode-se dizer que os nós i e j são incompatíveis se os seguintes caminhos são infactíveis:

- $\{i, j, n+i, n+j\}$;
- $\{i, j, n+j, n+i\}$;
- $\{j, i, n+i, n+j\}$;
- $\{j, i, n+j, n+i\}$;
- $\{i, n+i, j, n+j\}$;
- $\{j, n+j, i, n+i\}$.

Se os nós i e j são incompatíveis, então pode-se eliminar os seguintes arcos: (i, j), (j, i), $(n+i, j)$, $(j, n+i)$, $(n+j, i)$, $(i, n+j)$, $(n+i, n+j)$ e $(n+j, n+i)$.

Cortes:

Os cortes abaixo foram descritos no Capítulo 4 para o problema com frota heterogênea. Sendo assim, para os Modelos 4 e 5 os cortes descritos no Capítulo 4 são utilizados. Para o caso com a frota homogênea, isto é, Modelos 1, 2 e 3, os cortes dados abaixo são os acrescentados aos modelos:

1. Eliminação de sub-rota:

   - Para cada par de nós i,$j \in P$, os seguintes conjuntos S podem ser incorporados na restrição (3.17):
     - $S = \{i, j\} \Rightarrow x_{ij} + x_{ji} + x_{n+i,j} + x_{n+j,i} \leq 1$;
     - $S = \{i, n+j\} \Rightarrow x_{i,n+j} + x_{n+j,i} + x_{n+i,n+j} \leq 1$;
     - $S = \{i, n+i, j\} \Rightarrow x_{ij} + x_{ji} + x_{i,n+i} + x_{j,n+i} + x_{n+i,j} + x_{n+j,i} + x_{n+j,n+i} \leq 2$.
   - Para cada par de nós i,$j \in P$, os seguintes conjuntos S podem ser incorporados na restrição (3.18):
     - $S = \{n+i, n+j\} \Rightarrow x_{n+i,n+j} + x_{n+j,n+i} + x_{n+i,j} + x_{n+j,i} \leq 1$;
     - $S = \{i, n+j\} \Rightarrow x_{i,n+j} + x_{n+j,i} + x_{i,j} \leq 1$;
     - $S = \{i, n+i, n+j\} \Rightarrow x_{i,n+j} + x_{n+j,i} + x_{i,n+i} + x_{n+j,n+i} + x_{n+i,n+j} + x_{ij} + x_{n+i,j} \leq 2$.

- Similarmente, as inequações (3.19) e (3.20) geram as seguintes inequações:
  - $S = \{n+i, j, i\} \Rightarrow x_{n+i,j} + 2x_{j,n+i} + x_{j,i} + x_{i,n+i} + x_{n+j,n+i} \leq 2$;
  - $S = \{i, n+i, n+j\} \Rightarrow x_{i,n+i} + x_{n+i,n+j} + x_{n+j,i} + 2x_{i,n+j} + x_{i,j} \leq 2$.
- Para a inequação clássica de eliminação de sub-rota (3.16), temos o seguinte conjunto S e a seguinte inequação:
  - $S = \{i, j, n+i, n+j\} \Rightarrow x_{i,j} + x_{i,n+i} + x_{i,n+j} + x_{j,i} + x_{j,n+i} + x_{j,n+j} + x_{n+i,i} + x_{n+i,j} + x_{n+i,n+j} + x_{n+j,i} + x_{n+j,j} + x_{n+j,n+i} \leq 3$.

2. Restrição de Precedência (2.18): para cada par de nós i,$j \in P$, os seguintes casos particulares valem:
   - $S = \{s_0, i, n+j\} \Rightarrow x_{s_0,i} + x_{i,n+j} + x_{n+j,i} \leq 1$;
   - $S = \{i, n+j, en_0\} \Rightarrow x_{i,n+j} + x_{n+j,i} + x_{n+j,en_0} \leq 1$;
   - $S = \{s_0, i, n+i, n+j\} \Rightarrow x_{s_0,i} + x_{i,n+i} + x_{i,n+j} + x_{n+j,i} + x_{n+i,n+j} + x_{n+j,n+i} \leq 2$;
   - $S = \{i, j, n+j, en_0\} \Rightarrow x_{i,j} + x_{j,i} + x_{i,n+j} + x_{n+j,i} + x_{j,n+j} + x_{n+j,en_0} \leq 2$.

3. Restrições de ordem generalizadas (3.21): para cada par de coleta i,$j \in P$, a seguinte inequação é válida:
   - $x_{i,n+j} + x_{n+j,i} + x_{n+i,j} + x_{j,n+i} \leq 1$;

4. Restrição de Caminhos Infactíveis (3.26): para cada par de coletas i,$j \in P$, tal que $t_{ij} + d_j + t_{j,n+j} + d_{n+j} + t_{n+j,n+i} > L$, sendo L o fim da janela de tempo do depósito final, então,
$$x_{ij} + x_{j,n+j} + x_{n+j,n+i} \leq 1.$$

5. Para cada par de nós que são incompatíveis i,$j \in P$ e para cada nó $l \in P \cup D$, as seguintes inequações são válidas:
   - $x_{i,l} + x_{l,i} + x_{l,j} + x_{j,l} \leq 1$;
   - $x_{i,l} + x_{l,i} + x_{l,n+j} + x_{n+j,l} \leq 1$;
   - $x_{n+i,l} + x_{l,n+i} + x_{l,j} + x_{j,l} \leq 1$;
   - $x_{n+i,l} + x_{l,n+i} + x_{l,n+j} + x_{n+j,l} \leq 1$.

Pré-processamento específico para problema desse estudo:

Arcos:

1. Nós que sabemos previamente que o navio não pode atracar: se o navio k não pode atracar, por exemplo, em 5, 6 e 7 e o calado não está flexibilizado, podemos fixar $x_{k,5} = 0$, $x_{k,6} = 0$ e $x_{k,7} = 0$. Ou seja, se $A_{ik} = 1$ para algum $i \in N$ e para algum $k \in K$, então $x_{ki} = 0$;

2. O navio não sai do depósito final ou inicial e vai para o depósito de outro navio:

$$x_{ij} = 0, \ \ \forall i \in ST, j \in ST;$$

$$x_{ij} = 0, \ \ \forall i \in EN, j \in EN;$$

3. Nenhum navio entra nos depósitos iniciais:

$$x_{ij} = 0, \ \ \forall i \in N : i \neq s_0, j \in ST;$$

4. Nenhum navio sai do depósito final:

$$x_{ij} = 0, \ \ \forall i \in EN, j \in N : j \neq en_0;$$

5. Se i e j são o mesmo nó, então não existe caminho de i para j:

$$x_{ij} = 0, \forall i \in N, j \in N : i = j;$$

6. Não existe arco entre o nó de saída de um navio para um nó de entrega qualquer (desde que esta entrega não seja do nó artificial do navio), considerando que o navio comece vazio:

$$x_{ij} = 0, \ \ \forall i \in N, j \in D, k \in K, ord_i = s_k;$$

7. Não existe arco entre um nó de coleta qualquer (desde que esta coleta não seja a coleta artificial do navio) e um nó de chegada de um navio:

$$x_{ij} = 0, \ \ \forall i \in P, j \in N, k \in K, ord_j = e_k;$$

8. Não existe arco entre o depósito inicial e um nó de coleta ou entrega:

$$x_{s_0 j} = 0, \ \ \ \forall j \in N : j \neq ST;$$

9. Fixação de variáveis em relação ao posicionamento dinâmico (i é um navio e j uma plataforma):

$$x_{ij} = 0, \ \ \forall i \notin K_{DP}, j \in C_{DP}.$$

Cortes:

1. Se não se pode viajar de $i \in K$ para $j \in N$, pois $A_{ji} = 1$ e o $CF_{ji} \leq 0$, ou seja, o navio i não pode atracar em j e o calado não está flexibilizado, então temos o seguinte corte:

$$x_{i,l} + x_{lj} \leq 1, \quad \forall l \in N;$$

2. Se não se pode viajar de $i \in K$ para $j \in N$, pois $K_{DP_i} = 0$ e $C_{DP_j} = 1$, ou seja, o navio i não pode atracar em j, então,

$$x_{i,l} + x_{lj} \leq 1, \quad \forall l \in N;$$

3. O corte abaixo garante que o instante de início de serviço no nó de coleta i deve ser menor ou igual ao instante de início de serviço do nó de entrega $n + i$:

$$B_i + d_i + t_{i,n+i} \leq B_{n+i}, \quad \forall i \in P.$$

5.4 Métodos *branch-and-cut* propostos

Em síntese segue abaixo uma descrição dos métodos propostos, bem como modelos utilizados e quais desigualdades foram testadas e aplicadas em cada método.

1. Método 1: este método resolve o Modelo 1 sem as restrições (5.4), ou seja, tem como cortes obrigatórios (restrições que garantem a factibilidade da solução ótima) as restrições de precedência, impossibilidade de atracar, calado flexível e posicionamento dinâmico. Os outros cortes são colocados no método, mas com o intuito de melhorar seu limitante inferior;

2. Método 2: este método resolve o Modelo 2 (sem as restrições (5.4), (5.15) e (5.16)) e os algoritmos de separação obrigatórios são: precedência, caminhos infactíveis em relação às janelas de tempo, capacidade e caminhos infactíveis em relação à impossibilidade de atracar, calado flexível e posicionamento dinâmico. Os outros cortes como, por exemplo, de alcance e eliminação de sub-rota são gerados apenas para melhorar seu limitante inferior;

3. Método 3: resolve o Modelo 3, enquanto que as únicas restrições obrigatórias para a garantia de factibilidade são de caminhos infactíveis em relação à impossibilidade

de atracar, calado flexível e posicionamento dinâmico. Todos os demais cortes são incluídos, inclusive o de precedência, com o intuito de melhorar seu limitante inferior;

4. Método 4: este método proposto não possui nenhum corte que seja obrigatório para se ter a garantia de factibilidade. Este método resolve o Modelo 4 e todos os cortes são verificados a fim de, ao serem incluídos, melhorar seu limitante inferior;

5. Método 5: resolução do Modelo 5 e além disso, tem-se que as restrições em relação à impossibilidade de atracar, calado flexível e posicionamento dinâmico são obrigatórias para a garantia de factibilidade da solução ótima. Os demais cortes são incluídos para melhorar seu limitante inferior.

A Tabela 8 abaixo mostra quais cortes são obrigatórios para a garantia de factibilidade da solução ótima em cada método e quais são utilizados somente para a melhoria do seu limitante inferior.

Tabela 8: Resumo dos métodos propostos.

Métodos	Atrac./Calado/DP		Precedência		Capacidade		Sub-rota		Ordem Gener.		Cam. Infactív.		Alcance	
	Obri.	Melh.	Obri.	Melh.	Obri.	Melh.	Obri.	Melh.	Obri.	Melh.	Obri.	Melh.	Obri.	Melh.
1	X		X			X		X		X		X		X
2	X		X		X			X		X	X			X
3	X			X		X		X		X		X		X
4		X		X		X		X		X		X		X
5	X			X		X		X		X		X		X

5.5 Experimentos Computacionais

Esta seção mostra primeiramente os detalhes da implementação do método *branch-and-cut*, ou seja, as características do computador utilizado, funções implementadas, tempo limite, dentre outras. Depois, são mostrados os resultados computacionais juntamente com uma análise detalhada dos mesmos em que se compara os métodos *branch-and-cut* implementados com os resultados do modelo matemático apresentados na Seção 4.6.

5.5.1 Detalhes da Implementação

Para os testes realizados com as variações do método *branch-and-cut* aqui apresentados, são utilizados os mesmos conjuntos de dados reais definidos na Seção 4.5.2. Na implementação dos métodos, utilizam-se as funções do tipo *callback* da biblioteca *Concert*

do *solver* IBM CPLEX, que permitem a inserção de desigualdades válidas de forma integrada à resolução do problema. Cabe ressaltar que os cortes do próprio CPLEX ainda são inseridos com o uso da função *callback*. Os cortes foram acrescentados utilizando esta função, para que pudéssemos utilizar também os cortes do próprio CPLEX para melhoria do limitante inferior em cada método. Além disso, a *callback* é chamada apenas nos 10 primeiros nós da árvore e isto é feito por observar que nos níveis mais profundos da árvore não são gerados muitos cortes. Ainda, somente os cortes mais violados são acrescentados ao problema e limita-se em 100 cortes por desigualdade e por iteração.

A ramificação do método é feita pelo próprio *solver* CPLEX. Sendo assim, resolve-se a relaxação linear do modelo correspondente, aplica-se os algoritmos de separação correspondentes. Se a solução for fracionária, escolhe-se uma variável para que a árvore seja ramificada. Todas as decisões a respeito da ramificação são tomadas pelo *solver* CPLEX. O computador utilizado para resolver os exemplares é o mesmo descrito na Seção 4.5.1 e o tempo limite de processamento é novamente de 18.000 segundos.

Cabe mencionar que primeiramente o Modelo 4 foi executado sem a função *callback*, ou seja, sem adicionar as desigualdades válidas descritas na Seção 5.2. Os resultados mostram que o modelo de 3-índices do Capítulo 4 obtém resultados similares ao Modelo 4 executado utilizando o CPLEX e sem a inserção dos cortes descritos neste capítulo e com isto, utilizamos somente os resultados já apresentados no capítulo anterior.

5.5.2 Resultados Computacionais

As Tabelas 9 e 10 mostram os resultados para os casos de teste 1 e 2, respectivamente. Cada linha da tabela mostra o resultado de um método diferente e, além disso, a primeira linha em cada exemplar mostra o resultado obtido através do "modelo puro" (sem a adição dos cortes descritos na Seção 5.2) e dados no Capítulo 4. A coluna denominada UB mostra o valor da função objetivo, a próxima coluna mostra o *gap* dado em porcentagem (quando houver), a coluna denominada tempo mostra o tempo computacional total utilizado, no limite de 5 horas de processamento, a coluna denominada RL mostra qual o valor da relaxação linear de cada modelo e, por último, é mostrado quanto tempo o método levou para encontrar a melhor solução inteira. Os espaços em branco indicam que o método não encontrou solução factível no tempo estipulado e os espaços marcados pelo símbolo $-$ representam que o tempo máximo estipulado foi atingido.

Para o caso real 1, os métodos baseados nos Modelos 1 e 2 não foram bem sucedidos, uma vez que não encontram, ou encontram e não provam a solução ótima dos exemplares

Tabela 9: Resultados computacionais para o Caso 1.

Exem.	Métodos Propostos	UB	*Gap* (%)	Tempo (s)	RL	Tempo melhor sol.
$C1N10$	Modelo Puro Cap 4	1677,84	0%	6,49	1012,56	0,97
	Método 1	1677,84	0%	3261,06	850	115,19
	Método 2	1677,84	9,19%	–	850	536,58
	Método 3	1677,84	0%	226,84	850	5,64
	Método 4	1677,84	0%	10,76	1012,53	2,65
	Método 5	1677,84	0%	8,84	1012,52	1,35
$C1N15$	Modelo Puro Cap 4	2311,98	0%	20,62	1237,91	6,58
	Método 1	2313,37	17,94%	–	1200	16597,60
	Método 2	2311,98	21,48%	–	1200	16768,30
	Método 3	2313,37∗	15,70%	9372,73	1200	2516,44
	Método 4	2311,98	0%	16,17	1636,91	4,10
	Método 5	2311,98	0%	14,99	1587,42	3,10
$C1N20$	Modelo Puro Cap 4	2748,36	0%	2191,53	1364,30	278,77
	Método 1	2748,36	29,43%	–	1350	16523,20
	Método 2	2828,16∗	32,95%	13605,50	1350	13488,20
	Método 3	2748,36∗	24,49%	11407,70	1350	1129,37
	Método 4	2748,36	0%	24,19	2100,76	6,70
	Método 5	2748,36	0%	27,33	2069,51	10,18
$C1N25$	Modelo Puro Cap 4	3694,58	61,04%	–	1288,23	17887,42
	Método 1				1200	
	Método 2				1200	
	Método 3	3523,83	39,61%	–	1200	17398,30
	Método 4	3522,90	16,19%	–	1930,27	9172,70
	Método 5	3522,90	17,34%	–	1906,81	16408,00
$C1N30$	Modelo Puro Cap 4				1445,61	
	Método 1				1150	
	Método 2				1150	
	Método 3	4691,25	46,40%	–	1150	17996,90
	Método 4	4818,55	28,13%	–	2021,22	17922,80
	Método 5				1992,46	

* o método parou por limite de memória

Tabela 10: Resultados computacionais para o Caso 2.

Exem.	Métodos Propostos	UB	*Gap* (%)	Tempo (s)	RL	Tempo melhor sol.
$C2N10$	Modelo Puro Cap 4	1402,63	0%	16,34	699,72	6,85
	Método 1	1402,63	6,50%	–	1100	3705,09
	Método 2	1402,63	9,06%	–	1100	510,46
	Método 3	1402,63	1,79%	–	1100	893,54
	Método 4	1402,63	0%	16,38	1309,59	2,29
	Método 5	1402,63	0%	16,72	1309,59	3,71
$C2N15$	Modelo Puro Cap 4	1708,51	17,13%	–	607,31	125,11
	Método 1	1708,51	18,04%	–	950	17306,30
	Método 2	1708,51	18,88%	–	950	16385,10
	Método 3	1710,16*	15,51%	12128,70	950	1833,73
	Método 4	1708,51	0%	29,45	1225,55	9,22
	Método 5	1708,51	0%	24,78	1225,55	5,99
$C2N20$	Modelo Puro Cap 4	2452,90	47,71%	–	837,73	4094,45
	Método 1				1150	
	Método 2	2473,47	22,70%	–	1150	17924,40
	Método 3	2452,90	24,28%	–	1150	16285,60
	Método 4	2452,90	0%	445,33	1537,12	414,99
	Método 5	2452,90	0%	498,23	1565,97	467,90
$C2N25$	Modelo Puro Cap 4	3300,28	59,84%	–	1189,53	17396,19
	Método 1	3321,48	31,47%	–	1550	17864,00
	Método 2	3382,69	32,45%	–	1550	17837,50
	Método 3	3346,53	30,21%	–	1550	17965,60
	Método 4	3240,60	0,93%	–	2193,10	16069,90
	Método 5	3240,60	1,46%	–	2221,14	9517,99
$C2N30$	Modelo Puro Cap 4				1061,49	
	Método 1				1400	
	Método 2				1400	
	Método 3				1400	
	Método 4				1997,11	
	Método 5				1997,36	

* o método parou por limite de memória

$C1N10$, $C1N15$ e $C1N20$. A partir do $C1N25$, não encontram solução factível no tempo estipulado. O método baseado no Modelo 3 tem bom desempenho comparativamente aos demais, pois encontra solução factível para os exemplares $C1N25$ e $C1N30$. Para o $C1N30$ encontra melhor solução que o Método 4. Em relação aos métodos baseados nos Modelos 4 e 5 para os exemplares $C1N10$ e $C1N15$ não tem grandes melhorias, mas para o exemplar $C1N20$ encontram e provam a solução ótima em aproximadamente 20 segundos, enquanto que o modelo puro prova a solução ótima em 2191 segundos. Para o $C1N25$ encontram solução de melhor qualidade e com *gap* inferior ao encontrado pelo modelo puro, com o método baseado no Modelo 4 com uma leve vantagem no *gap* do exemplar $C1N25$. Temos que algumas soluções são ótimas apesar dos *gaps* positivos, como é o caso dos exemplares $C1N10$ e $C1N15$ para o Método 2 e do exemplar $C1N20$ para os métodos baseados nos Modelos 1 e 3. Cabe dizer que nenhum método foi capaz de encontrar solução factível pra nenhum exemplar de tamanho maior que $C1N30$.

Para o caso real 2, o desempenho dos Métodos 4 e 5 foram os melhores. Nestes métodos, para os exemplares $C2N15$ e $C2N20$, encontraram e provaram a solução ótima em poucos segundos, o que não ocorreu com nenhum outro método. Para o exemplar real $C2N25$, os métodos terminam com a mesma solução e com *gap* inferior a 2%, enquanto que o modelo puro não encontrou a melhor solução dos métodos em 5 horas de processamento. Os métodos com os Modelos 1, 2 e 3 não são bem sucedidos, uma vez que para a maioria dos exemplares o *gap* não é zerado e além disso, para alguns exemplares não é encontrado solução factível no tempo estipulado. Entretanto, podemos observar que em função dos resultados com os Métodos 4 e 5 pode-se concluir que esses *gaps* são bem menores, uma vez que algumas soluções são as ótimas, como é o caso do $C2N10$ para os Métodos 1, 2 e 3, do $C2N15$ para os métodos baseados nos Modelos 1 e 2 e o modelo puro e do $C2N20$ para o método baseado no Modelo 3 e o modelo puro.

Para exemplificar o tamanho do problema em cada um dos métodos propostos baseados nos diferentes modelos, considere o exemplar $C2N25$. A Tabela 11 mostra o número de variáveis e de restrições em cada abordagem proposta. Note que o número de variáveis é relativamente menor para os modelos com variáveis de 2-índices, enquanto que o número de restrições é consideravelmente maior em relação aos modelos com variáveis de 3-índices. Observe que o número de variáveis para os Modelos 4 e 5 é o mesmo, considerando que o modelo é igual, somente com algumas restrições a menos (por isso que o Modelo 5 possui menos restrições comparado ao Modelo 4).

Note que os modelos que possuem variáveis com 2-índices apresentam uma modelagem

Tabela 11: Número de variáveis e restrições em cada modelo para o $C2N25$.

Modelos	# Variáveis	# Restrições
Cap 4	415.189	398.550
1	13.964	1.088.187
2	13.736	1.055.243
3	14.078	1.113.837
4	508.898	865.181
5	508.898	861.154

mais compacta, porém ocorre a perda de algumas restrições, devido à falta do índice k. Então, tem-se modelos mais compactos e menores, entretanto mais fracos em relação à relaxação linear.

A Figura 37 abaixo mostra a variação dos limitantes inferior e superior ao longo do tempo para o exemplar $C2N25$ para o Método 4. A melhor solução encontrada é 3240,60 em 5 horas de processamento com *gap* de 0,93%. O gráfico ilustra que o limitante inferior se inicia com valor perto de 2000, dá um pequeno salto depois de 40 segundos e, em seguida, cresce quase que linearmente. Já a primeira solução inteira encontrada foi perto dos 240 segundos e teve valor igual a 8222,24.

Em relação ao desempenho dos cortes na melhoria do valor do limitante inferior, os que mais apresentam impacto são os cortes do tipo alcance. Os cortes mais gerados por todas os exemplares são do tipo alcance e precedência. Em relação à relaxação linear de cada modelo, os que apresentam a relaxação mais apertada são os Modelos 4 e 5. Já os modelos com variáveis de 2-índices (Modelos 1, 2 e 3) apresentam relaxação linear muito similar entre si.

A Figura 38 mostra as rotas de dois navios na solução do $C2N25$ para o Método 4. O navio 14 possui uma capacidade de 116450 e visita as plataformas 54 e 48 que são diferentes, então note que este navio opta por coletar e entregar ao invés de coletar e coletar, já que assim, não paga o custo por visita consecutiva. Com relação à rota do navio 24, este possui capacidade de 168756 e visita as plataformas 33, 34, 35 e 36 que representam a mesma plataforma, com janelas de tempo e demandas diferentes. Assim, não há custo por visita consecutiva. O mesmo ocorre com os terminais 89, 90, 91 e 92. A solução ótima deste exemplar é de 3240,60 unidades. Deste total, 990,60 são relacionados ao custo das viagens, 2250 em relação às atracações e 0 em relação à visita consecutiva a plataformas diferentes.

Cabe fazer uma comparação com os resultados obtidos na literatura para problemas

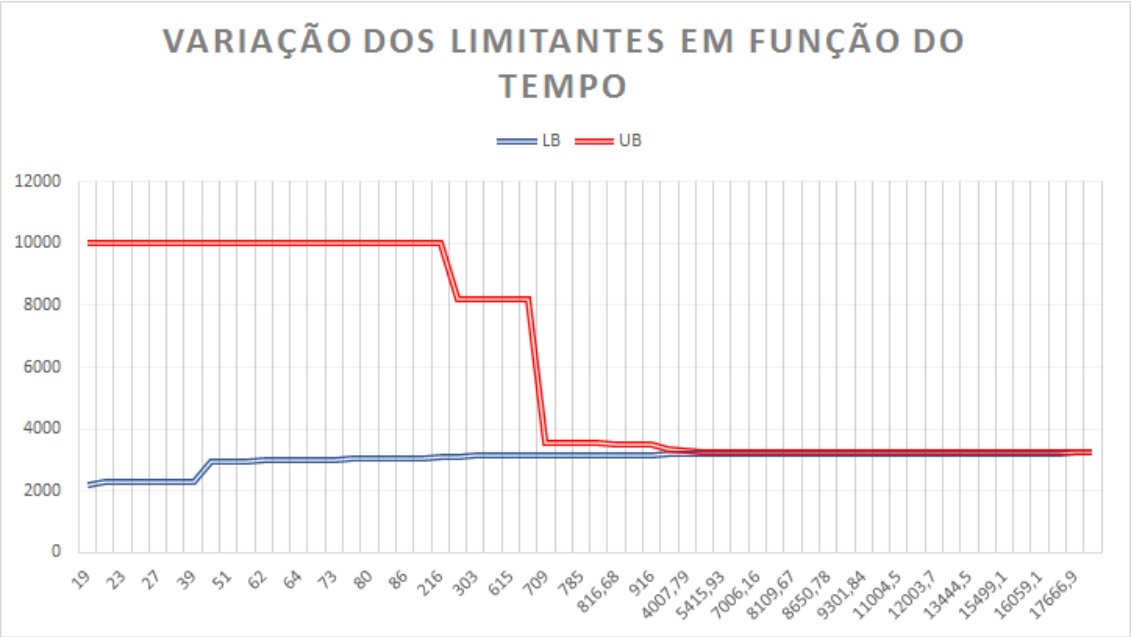

Figura 37: Gráfico que mostra a variação dos limitantes para o $C2N25$ para o Método 4.

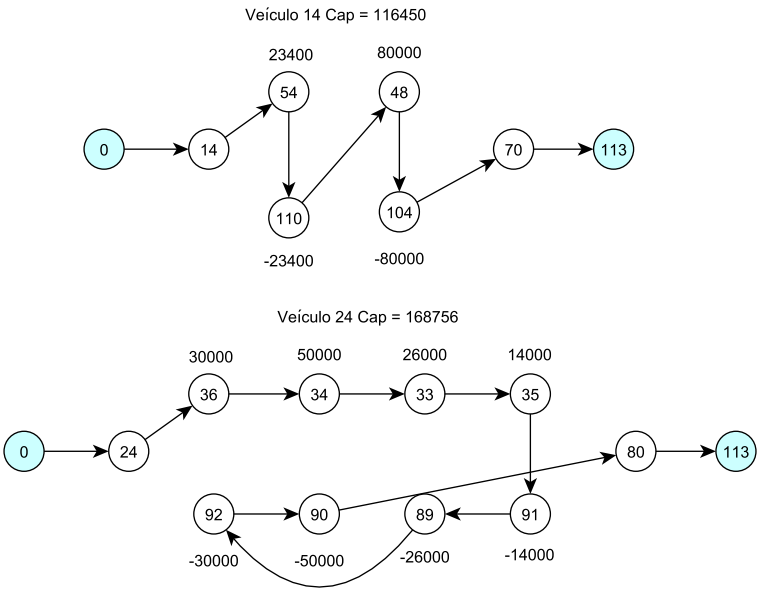

Figura 38: Rotas de dois navios na solução do $C2N25$.

de coleta e entrega (ROPKE; CORDEAU, 2009) com os métodos aqui propostos para o problema de coleta e entrega desse estudo de caso. Os melhores resultados na literatura foram encontrados com os métodos baseados nos Modelos 1 e 2, o que não ocorreu aqui para esse problema específico com os exemplares reais fornecidos pela empresa, em que os melhores resultados foram encontrados com métodos baseados nos Modelos 3, 4 e 5. O Método 5 melhorou o tempo e provou a solução ótima para alguns casos e os Modelos 3 e 4 encontraram solução factível para o exemplar de 30 pares de coletas e entregas, os quais os outros métodos não encontraram.

Testes adicionais foram realizados em cenários em que as janelas de tempo dos exemplares foram aumentadas em 10%, 20%, 50% e 100%. Entretanto, os resultados mostraram que as soluções não melhoraram substancialmente, conforme era esperado. Sendo assim, tais resultados não foram reportados nesta tese.

Diante desses resultados, pode-se fazer algumas observações. Estes métodos possuem uma limitação para resolver problemas com mais de 25 pares de coleta e entrega, o que corresponderia a cerca de 10 dias de operação na empresa. Do ponto de vista prático essas abordagens não seriam aceitáveis, uma vez que os operadores dificilmente ficariam 5 horas esperando por uma solução viável.

6 *Método* branch-and-price *para o problema da indústria petrolífera*

Este capítulo tem o objetivo de apresentar todas as características do método *branch-and-price* desenvolvido para resolver especificamente o problema de coleta e entrega e janelas de tempo da indústria de petróleo. Primeiramente, o problema é formulado como um modelo de particionamento de conjuntos usado para resolver o modelo proposto, incluindo a técnica de geração de colunas, algoritmos para solução dos subproblemas e regras de ramificação. Por fim, os resultados computacionais são mostrados juntamente com uma análise detalhada dos mesmos.

6.1 Particionamento de Conjuntos

Seja Ω_k o conjunto de todas as rotas factíveis do navio $k \in K$. Uma rota factível satisfaz todas as restrições do modelo dado na Seção 4.3, exceto as restrições que garantem que todos os nós sejam visitados e que cada nó seja visitado por um único navio, ou seja, uma rota factível garante que a capacidade do veículo seja respeitada, janelas de tempo, restrições de fluxo, que cada veículo sai de seu depósito específico e retorna ao seu depósito específico, restrições temporais, que a coleta seja feita antes da respectiva entrega, restrições de atracação, calado flexível e posicionamento dinâmico. Portanto, uma rota factível satisfaz todas as restrições (4.4)-(4.26).

Para cada rota $r \in \Omega_k$, seja c_r o custo desta rota, tendo em conta o custo das viagens entre os nós, custo por atracação e penalização por visita consecutiva a plataformas diferentes (custos definidos na função objetivo (3.1)). Seja a_{ir} a constante que indica o número de vezes em que o nó $i \in P$ é visitado pela rota r, ou seja, cada coluna $a_r = (a_{r1},...,a_{rn})^T$ é um vetor binário em que $a_{ir} = 1$ se e somente se, a rota r visita o nó i. Seja y_r uma variável binária que assume valor 1 se, e somente se, a rota $r \in \Omega_k$, para algum $k \in K$ é utilizada na solução do problema. Todas as restrições (4.4)-(4.26) estão impostas impli-

citamente na formulação dada a seguir de particionamento de conjuntos, uma vez que as rotas Ω_k satisfazem todas estas restrições. O problema é formulado da seguinte maneira e chamado de Problema Mestre (PM) para frota heterogênea, conforme Seção 3.3:

$$\text{Min} \sum_{k \in K} \sum_{r \in \Omega_k} c_r y_r \tag{6.1}$$

s.a:

$$\sum_{k \in K} \sum_{r \in \Omega_k} a_{ir} y_r = 1 \qquad \forall i \in P \tag{6.2}$$

$$\sum_{r \in \Omega_k} y_r \leq 1 \qquad \forall k \in K \tag{6.3}$$

$$y_r \in \{0,1\} \qquad \forall k \in K, r \in \Omega_k \tag{6.4}$$

A função objetivo (6.1) modela o custo das rotas utilizadas, enquanto que as restrições (6.2) garantem que todo nó é servido exatamente uma vez. As restrições (6.3) limitam que apenas um navio irá atender à rota r. As restrições (6.4) garante o domínio da variável y_{rk}. A formulação (6.1)-(6.4) pode ser obtida aplicando a decomposição de *Dantzig-Wolfe* (DANTZIG; WOLFE, 1960), conforme discutido no Capítulo 3.

Um limitante inferior na solução de (6.1)-(6.4) pode ser obtido resolvendo a relaxação linear do modelo, ou seja, substituindo (6.4) por

$$y_r \geq 0, \quad \forall k \in K, r \in \Omega_k. \tag{6.5}$$

Devido ao número elevado de variáveis é difícil resolver essa relaxação linear. Para tanto, utiliza-se do método de geração de colunas para resolver tal relaxação. Uma vez que temos variáveis binárias, podemos combinar o método de geração de colunas com o método *branch-and-bound*, resultando no *branch-and-price*. A geração de colunas é um processo iterativo que resolve um problem auxiliar, chamado de problema mestre restrito (PMR), em que este é obtido considerando um subconjunto relativamente pequeno $\overline{\Omega} \subseteq \Omega_k$ de rotas disponíveis.

Seja $\pi_i \in \mathbb{R}$ a variável dual associada à restrição (6.2) para o nó $i \in P$ e $\mu \in \mathbb{R}$ a variável dual associada à restrição (6.3) reescrita como uma inequação $\geq$. O custo relativo

de cada coluna (rota) r é dado por:

$$\widetilde{c}_r = c_r - \sum_{i \in P} \pi_i a_{ir} + \mu_k$$

A cada iteração do método de geração de colunas, resolve-se o PMR para se obter uma solução dual $(\widetilde{\pi},\widetilde{\mu})$, que pode ser utilizada para gerar colunas que ainda não foram geradas pelo PMR. Estas colunas estão associadas com rotas factíveis, as quais são geradas resolvendo o seguinte subproblema (SP) (*pricing*), para todo $k \in K$:

$$z_{SP}^{k}(\widetilde{\pi},\widetilde{\mu}) = \text{Min}\left\{ c_r - \sum_{i \in P} \pi_i a_{ir} + \mu_k | r \in \Omega_k \right\} \tag{6.6}$$

Se o valor correspondente do custo relativo $z_{SP}^{k}(\widetilde{\pi},\widetilde{\mu})$ for negativo, então uma nova variável y_r pode ser adicionada ao PMR utilizando esta nova rota r. Uma vez que a rota r é adicionada ao $\overline{\Omega}$, o PMR é resolvido novamente. Se $z_{SP}^{k}(\widetilde{\pi},\widetilde{\mu}) \geq 0$, então a rota correspondente ao veículo k é descartada. Se $z_{SP}^{k}(\widetilde{\pi},\widetilde{\mu})$ é não-negativo para todo $k \in K$, então $(\widetilde{\pi},\widetilde{\mu})$ é solução ótima dual do respectivo PMR. Logo, a solução ótima do PMR atual é também solução ótima da relaxação do problema mestre e consequentemente, o método de geração de colunas termina. Além disso, o custo c_r e as variáveis duais μ_k dependem do navio k utilizado, então a cada iteração da geração de colunas tem-se que resolver $|K|$ subproblemas. Nas próximas seções, descrevemos como resolver cada subproblema.

6.2 Problema de Caminho Mínimo Elementar

O subproblema, definido anteriormente, é chamado de problema de caminho mínimo elementar, em que não se permite sub-rotas na solução ótima e normalmente é resolvido utilizando-se um algoritmo de programação dinâmica, conhecido como *label-extension* (ROPKE; CORDEAU, 2009; BETTINELLI et al., 2014). O problema de caminho mínimo restrito consiste em decidir a melhor rota em um grafo $G = (N,A)$ ao sair de um nó fonte s (depósito inicial s_0) e chegar em um destino t (depósito final en_0). Cada arco do grafo está associado a um custo e o custo do caminho é dado pela soma dos custos dos arcos utilizados no caminho. Os caminhos devem satisfazer diversas restrições como, por exemplo, tempo, capacidade do navio, calado flexível, posicionamento dinâmico, dentre outras. Para considerarmos frota heterogênea, temos que resolver um subproblema para cada navio e manter o índice k do navio guardado em todas as rotas geradas, para sabermos qual navio foi utilizado em cada rota específica (Ω_k). Note que uma rota factível para um

dado navio pode não ser factível para outro navio.

Os algoritmos de *label* constroem caminhos parciais do grafo G. Cada caminho parcial inicia sua rota em s e termina sua rota em um determinado nó $i \in N$. Caminhos parciais são estendidos através de arcos do grafo e primeiramente todos os *labels* partem do depósito s_0.

A fim de acelerar a programação dinâmica, podemos descartar *labels* (caminhos parciais) através de regras de dominância. Um caminho p domina outro caminho p' se ambos estão no mesmo nó $i \in N$, o custo de p é menor ou igual ao custo de p', e todas as extensões factíveis de p' pelo caminho parcial até o nó final t, também são factíveis em p. Não é viável verificar se um caminho domina outro através desta definição. Então existem regras, que são condições suficientes, mas não necessárias, para checar se um caminho domina outro e eliminá-lo da programação dinâmica. Estas regras são descritas na Subseção 6.2.5.

O algoritmo de *label* descrito aqui é baseado no algoritmo descrito por Ropke e Cordeau (2009), o qual é menos restrito que os propostos por Sol (1994) e Sigurd et al. (2004) para o problema de coleta e entrega e janelas de tempo. Um algoritmo menos restrito consiste na eliminação de mais *labels*, ou seja, em um algoritmo mais rápido computacionalmente

Em cada *label* são guardadas as seguintes informações: o nó em que o label está, digamos que seja o nó η, a carga l do navio após a visita em η, o tempo t de início de serviço em η, o custo acumulado c, o conjunto O de requisições em que o nó de coleta foi visitado e em que o nó de entrega não foi visitado. Além disso, é guardado o conjunto U de nós inalcançáveis (*unreachable*), ou seja, o nó de coleta foi visitado, porém não é possível que o nó de entrega seja visitado, ou o caminho de η até o nó pertencente a U viola as janelas de tempo. Sendo assim, em cada *label* ficam armazenados os recursos η, t, l, c, U e O.

A notação $t\,(L)$ é usada pra referenciar o tempo de início de serviço no nó η do *label* L e o mesmo é feito com outros recursos. Quando um *label* é estendido, precisa-se verificar todos os seus recursos, ou seja, verificar capacidade, janelas de tempo, atracações, calado flexível e posicionamento dinâmico, além disso, o custo precisa ser atualizado a cada novo *label*. Por exemplo, se vamos estender um *label* do nó i para um nó j para o recurso capacidade, guardamos a informação do quanto foi coletado ou entregue ao longo do *label* até o nó i, e o novo *label* será factível se a quantidade carregada ou descarregada não exceder a capacidade do navio. Para o recurso janela de tempo, também guarda-se a

informação do tempo total decorrido ao longo do *label*. Para que o novo *label* seja factível depois de inserir o nó j, precisa-se verificar se o tempo decorrido até o nó i, mais o tempo de duração de serviço em i, mais o tempo de viagem entre i e j, não ultrapasse o fim da janela de tempo em j. O mesmo é feito para os outros recursos.

Em Righini e Salani (2006), os autores propuseram desenvolver a programação dinâmica de forma bidirecional a fim de melhorar o desempenho da mesma. Neste tipo bidirecional, cada *label* é estendido para frente do vértice s (fonte), ou seja, para seus sucessores (progressiva - *forward*), e para trás a partir do vértice t (destino), para seus antecessores (regressiva - *backward*). Um caminho é dado juntando dois caminhos parciais, um feito de s até um dado nó i e outro caminho do mesmo nó i até t. O primeiro trabalho a tratar da programação dinâmica de forma bidirecional para o problema de coleta e entrega foi em Bettinelli et al. (2014). Os autores propuseram um algoritmo *branch-and-price* para o problema de coleta e entrega com frota heterogênea, múltiplos depósitos e janelas de tempo que podiam ser violadas. Além disso, os autores destacaram que a programação dinâmica feita desta maneira não é trivial, uma vez que precisa-se ter cuidado ao trabalhar com os pares de coleta e entrega nos caminhos progressivo e regressivo.

As seções que se seguem definem e explicam em detalhes o algoritmo de *label*, juntamente com as estratégias utilizadas para resolver a programação dinâmica, como as regras de extensão progressiva e regressiva, e também como juntar estes *labels* para resultar em um caminho que seja factível de acordo com todos os recursos disponíveis. Além disso, mostram as regras de dominâncias e também as heurísticas utilizadas para gerar rotas com custo negativo para o problema mestre ou melhorar as rotas já existentes.

6.2.1 Extensão do *Label*

Descrevemos nesta subseção como ocorre a extensão do *label*, ou seja, como é o procedimento para estender o caminho de forma progressiva (de s_0 até um nó i) e de forma regressiva (de en_0 até i).

Extensão Progressiva (*Forward*)

Aqui descrevemos como estender um *label* do nó inicial s_0 até um dado nó i. O *label* é estendido até o meio do horizonte de tempo, ou seja, $T_{max}/2$, sendo T_{max} dado pelo fim do horizonte de tempo. A extensão do *label* é feita até que pelo menos metade de um dos seus recursos tenham sido gastos como, por exemplo, o recurso tempo.

A extensão de um *label* L para um nó j, ou seja, $(\eta(L), j)$, é possível se $\eta(L) \neq j$, $t(L) + t_{\eta(L),j} \leq b_j$ e $l(L) + q_j \leq Q$, ou seja, se garantir que o nó j não está no *label*, que as janelas de tempo e a capacidade do navio sejam respeitadas. Precisamos garantir também as questões relacionadas a atracações, calado flexível e posicionamento dinâmico. Em relação a calado flexível e posicionamento dinâmico, a verificação é feita por meio do recurso carga l. Por exemplo, se o navio possui DP e temos a plataforma j para extensão do *label*, então essa extensão somente é possível se a quantidade de carga l for igual a no máximo 50% da capacidade do navio. Cabe ressaltar que a questão de atracar é tratada eliminando arcos do grafo. Como não depende da carga acumulada do navio no momento da atracação, podemos retirar o arco correspondente ao navio k atracar em um ponto operacional i, por exemplo. Desta maneira, se para alguma rota $r \in \Omega_k$ de algum navio $k \in K$, este não puder atracar em i, então na coluna r teremos o valor $a_{ir} = 0$.

Além disso, a cada nó verificado, o tempo de serviço será contabilizado somente se os nós forem plataformas ou terminais diferentes (igual é feito no modelo matemático). Por exemplo, na Figura 39 temos que o *label* será estendido do nó i para o nó j. Se estes nós representarem o mesmo ponto operacional, então o tempo de serviço não é contabilizado no nó i; se os nós representarem pontos operacionais diferentes, então o tempo de serviço será contabilizado no nó i.

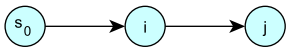

Figura 39: Exemplo da extensão progressiva do *label*.

Além disso, L e j devem satisfazer uma das seguintes condições, o que garante que a extensão seja compatível:

1. $0 < j \leq n \wedge j \notin U(L)$;
2. $n < j \leq 2n \wedge j - n \in O(L)$.

Ou seja, se o nó j é um nó de coleta, então este não pode estar contido no conjunto U; se j é um nó de entrega, então a coleta respectiva precisa estar no conjunto O (garantimos assim, a precedência); e se o nó j for o nó final en_0, então o conjunto O precisa estar vazio.

Extensão Regressiva (*Backward*)

Aqui descrevemos como estender um *label* do nó en_0 até um dado nó i. A situação inicial do nó é dada pelo fim do horizonte de tempo, ou seja, por T_{max}. A extensão de um *label* L para o nó j, ou seja, $(\eta(L), j)$ é possível se $\eta(L) \neq j$, $t(L) + t_{j,\eta(L)} \leq T_{max} - a_j$ e $l(L) + q_j \geq -Q$, ou seja, só é possível se garantir que as janelas de tempo e a capacidade do navio sejam respeitadas. Também precisamos garantir as questões de atracações, calado flexível e posicionamento dinâmico. Para as questões de calado flexível e posicionamento dinâmico, novamente utilizamos do recurso carga l para a verificação. Isto é realizado da mesma maneira que a extensão progressiva, porém neste caso, a capacidade do navio é colocada com o sinal invertido, já que a verificação é feita do fim para o começo e as demandas dos nós de entrega são negativas. Por exemplo, considere a Figura 40, o *label* iniciou sua rota no depósito final en_0, se deslocou para o nó i e agora, queremos verificar se a extensão ao nó j é factível em relação à calado flexível e posicionamento dinâmico. Para sabermos qual a carga acumulada quando o navio chega ao nó j, precisamos saber se os nós i e j são coletas ou entregas. Suponha que i e j sejam entregas, então a carga acumulada do navio ao chegar em j é a soma das demandas de $i - n$ e $j - n$. Definido l, verificamos se é possível a extensão em relação à calado flexível e posicionamento dinâmico.

A cada *label* estendido precisamos contabilizar corretamente o tempo de serviço. Para isso, considere a Figura 40, nela vemos que o nó i será estendido ao nó j. A partir do depósito final (en_0), o tempo de serviço não será contabilizado no nó i, pois o tempo de serviço no depósito final é zero. Ao chegar ao nó j, temos que analisar se o tempo de serviço será contabilizado referente ao nó i. Este será positivo se a distância entre o nó pai (i) e o pai do pai (en_0) for maior que zero, ou o nó que representa o pai do pai (en_0) for o depósito final (o que ocorre na figura). Esta análise do tempo de serviço para o caso progressiva é direta, uma vez que apenas temos que considerar se a distância entre i e j é positiva. Já neste caso, na extensão regressiva esta análise não é direta, uma vez que estamos no sentido contrário do caminho, e precisa-se ter cuidado para que o tempo de serviço não seja contabilizado mais vezes.

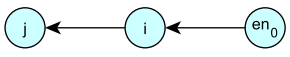

Figura 40: Exemplo da extensão regressiva do *label*.

Além disso, L e j devem satisfazer uma das seguintes condições, o que garante que a extensão seja compatível. Note que a definição do conjunto O para a extensão regressiva é outra: é o conjunto cujas entregas foram visitadas, mas as respectivas coletas não. Sendo assim, temos que:

1. $0 < j \leq n \wedge j + n \in O(L)$;
2. $n < j \leq 2n \wedge j - n \notin U(L)$.

Ou seja, se j é um nó de coleta, então a respectiva entrega precisa ter sido visitada (garantimos assim, a precedência). Se j é um nó de entrega, então a respectiva coleta não pode pertencer ao conjunto U.

6.2.2 Atualização do Label

Uma vez que a extensão do *label* foi possível de ser realizada, então um novo *label* deve ser criado, conforme nesta subseção. Novamente, temos que as atualizações progressiva e regressiva são feitas de maneiras diferentes. O custo atualizado é calculado da mesma maneira que a função objetivo do modelo matemático (4.1) apresentado anteriormente na Seção 4.3, ou seja, custo das viagens, custo por atracação e penalização por visita consecutiva a plataformas diferentes e é denotado por $d_{\eta(L),j}$, isto é, custo da viagem entre $\eta(L)$ e o novo nó j. Na definição deste custo d está incluída também a variável dual associada à restrição do modelo de particionamento de conjuntos (6.2).

Progressiva (*Forward*)

Depois de verificado que é possível estender o nó j de um *label* L, um novo *label* L' é criado da seguinte maneira:

1. $\eta(L') = j$;
2. $t(L') = max\left\{a_j, t(L) + t_{\eta(L),j}\right\}$;
3. $l(L') = l(L) + q_j$;
4. $c(L') = c(L) + d_{\eta(L),j}$;
5. $U(L') =$
   - $U(L) \cup \{j\}$, se $j \in P$;

- $U(L)$, se $j \in D$.

6. $O(L') =$

   - $O(L) \cup \{j\}$, se $j \in P$;
   - $O(L) \setminus \{j - n\}$, se $j \in D$.

Regressiva (*Backward*)

Depois de verificado que é possível estender o nó j de um *label* L, um novo *label* L' é criado da seguinte maneira:

1. $\eta(L') = j$;
2. $t(L') = max\left\{T_{max} - b_j, t(L) + t_{\eta(L),j}\right\}$;
3. $l(L') = l(L) + q_j$;
4. $c(L') = c(L) + d_{\eta(L),j}$;
5. $U(L') =$

   - $U(L) \cup \{j\}$, se $j \in D$;
   - $U(L)$, se $j \in P$.

6. $O(L') =$

   - $O(L) \cup \{j\}$, se $j \in D$;
   - $O(L) \setminus \{j + n\}$, se $j \in P$.

6.2.3 Conjunto Inalcançável

O conceito de conjunto inalcançável (*unreachable*) foi primeiramente proposto por Feillet et al. (2004), em que os autores propuseram um algoritmo exato para o problema de caminho mínimo elementar com limitação de recursos. No contexto do problema de coleta e entrega, o conjunto U (*unreachable*) é formado pelos nós que não conseguem ser visitados por aquele determinado *label*. Por exemplo, o nó de coleta j foi visitado, mas não é possível visitar o nó de entrega $(j+n)$ pelas janelas de tempo; ou estender este *label* para o nó de coleta j violaria as janelas de tempo.

A verificação de se um nó pertence ao conjunto U é feita diferentemente para os caminhos construídos de maneira progressiva e regressiva. Além disso, a verificação é feita a partir de um dado nó i para um nó j, isto é, verifica-se apenas para dois nós a fim de não consumir muito tempo computacional nesta fase.

Progressiva (*Forward*)

Temos que verificar algumas situações como, por exemplo, se o nó i é um nó de coleta ou entrega. O mesmo vale para o nó j. Se alguma das possibilidades abaixo for violada, então o nó j entra no conjunto $U\,(L)$. A notação $i \Rightarrow j$ indica que a verificação das janelas de tempo precisa ser feita do nó i para o nó j. Sendo assim, temos as seguintes possibilidades:

1. Se i e j são nós de coleta (PP) ou se i é um nó de entrega e j é um nó de coleta (DP):

   Verificar se $i \neq j$ (caso PP) e verificar as janelas de tempo de $i \Rightarrow j$;

2. Se i é um nó de coleta e j é um nó de entrega (PD) - j não pode ser a entrega respectiva da coleta i:

   Verificar a janelas de tempo de $i \Rightarrow j - n \Rightarrow j$;

3. Se i e j são nós de entrega (DD):

   Se $i \neq j$ e verificar janelas de tempo para o caminho $i \Rightarrow j - n \Rightarrow j$.

Regressiva (*Backward*)

A verificação neste caso é diferente, sendo assim temos as seguintes possibilidades:

1. Se i e j são nós de entrega (DD) ou se i é um nó de coleta e j é um nó de entrega (PD):

   Verificar se $i \neq j$ (caso DD) e verificar as janelas de tempo do caminho $i \Rightarrow j$;

2. Se i é um nó de entrega e j é um nó de coleta (DP) - j não pode ser a coleta respectiva da entrega i:

   Verficar a janelas de tempo de $i \Rightarrow j + n \Rightarrow j$;

3. Se i e j são nós de coleta (PP):

   Se $i \neq j$ e verificar janelas de tempo para o caminho $i \Rightarrow j + n \Rightarrow j$.

6.2.4 Junção de caminhos: Progressiva e Regressiva

Depois de obter todos os caminhos através das formas progressiva e regressiva, precisa-se juntar tais caminhos parciais de modo a obter um caminho completo do nó s_0 ao nó en_0, o qual respeite todas as restrições (4.4)-(4.26). Para a junção de um caminho parcial progressivo com um caminho parcial regressivo é preciso verificar algumas situações:

- Ambos os caminhos parciais precisam terminar exatamente no mesmo nó η;
- Nenhum nó pode estar presente em ambos os caminhos parciais, exceto o nó η (nó repetido);
- Precedência: para os nós de coleta que estão no caminho parcial progressivo cujos nós das respectivas entregas não estão em no caminho progressivo, precisa-se verificar se estas entregas estão no caminho parcial regressivo. Para as entregas que estão no caminho parcial regressivo cujas coletas respectivas não estão no caminho regressivo, precisa-se verificar se estas coletas estão no caminho parcial progressivo;
- Verificar capacidade e janelas de tempo.

6.2.5 Critérios de Dominância

Como já mencionado, existem critérios de dominância para que as extensões progressiva e regressiva não sejam feitas na totalidade, para todos os casos. Nestes critérios, se um *label* L_1 dominar outro *label* L_2, então o *label* L_2 será descartado e não mais estendido. Isto faz com que o algoritmo seja mais eficaz computacionalmente. As regras de dominância descritas abaixo foram propostas por Ropke e Cordeau (2009) para o problema de coleta e entrega e janelas de tempo. Sendo assim, um *label* L_1 domina outro *label* L_2 se:

- $\eta\left(L_1\right) = \eta\left(L_2\right)$;
- $t\left(L_1\right) \leq t\left(L_2\right)$;
- $c\left(L_1\right) \leq c\left(L_2\right)$;
- $U\left(L_1\right) \subseteq U\left(L_2\right)$;
- $O\left(L_1\right) \subseteq O\left(L_2\right)$.

Este critério de dominância é válido somente se a matriz de custo satisfizer a propriedade da desigualdade triangular de entrega (ROPKE; CORDEAU, 2009). Ou seja, se a matriz de custo for indicada por d_{ij}, então se $d_{ij} + d_{jk} \geq d_{ik}$ se j é um nó de entrega, então a matriz satisfaz a desigualdade triangular. Dizemos que uma matriz de custo que satisfaz essa propriedade satisfaz também a propriedade de *desigualdade triangular de entrega* (*delivery triangle inequality*). Como a implementação foi realizada de forma bidirecional, então esta propriedade tem que ser válida também para os nós de coleta, pela parte regressiva (BETTINELLI et al., 2014).

Este critério de dominância também é válido para o nosso caso específico. Considere primeiro o caso do caminho parcial progressivo em que dois *labels* estejam em um mesmo nó i, como mostra a Figura 41. Para o *label* L_1 dominar o *label* L_2, a quantidade de carga a bordo em L_1 precisa ser menor ou igual a quantidade de carga em L_2 ao chegar ao mesmo nó i. Além disso, temos que cada subproblema é resolvido especificamente para um navio, ou seja, as duas rotas da figura abaixo dizem respeito ao mesmo navio. Temos que verificar que ao dominar um *label*, digamos o L_2, este não encontraria uma rota com menor custo, considerando atracação, calado flexível ou posicionamento dinâmico. A questão da atracação é fixada no problema, uma vez que não depende da quantidade de carga a bordo do navio. Em relação ao calado flexível e posicionamento dinâmico, estes dependem da quantidade a bordo do navio. Qualquer que seja a rota encontrada pelo *label* L_2, o *label* L_1, conseguiria encontrar a mesma rota, uma vez que a quantidade a bordo do nó i em L_1, é menor ou igual a L_2 e ambos os *labels* se referem ao mesmo navio. Sendo assim, o critério de dominância é válido também considerando as questões do problema do estudo de caso (atracação, calado flexível e posicionamento dinâmico).

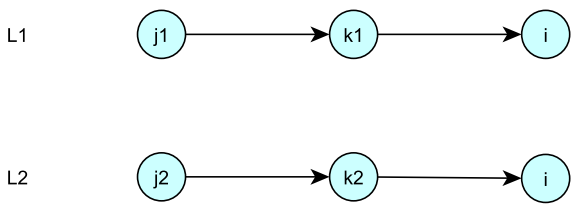

Figura 41: Exemplo do critério de dominância para o caminho progressivo.

A explicação do porquê este critério de dominância em relação ao caminho regressivo é válido também para o nosso caso específico é análoga a explicação já apresentada, ou seja, o critério de dominância apresentado é também válido considerando as questões de atracações, calado flexível e posicionamento dinâmico. Considere dois *labels* que estejam em um mesmo nó i (Figura 42). A explicação se deve ao fato de que ambos os *labels* se referem ao mesmo navio e que para o *label* L_1 dominar o *label* L_2, a quantidade de carga a bordo em L_1 precisa ser menor ou igual a quantidade de carga em L_2 ao chegar ao mesmo nó i. Então, qualquer que seja o caminho que o L_2 encontrar, o *label* L_1 também encontraria com custo menor ou igual a L_2.

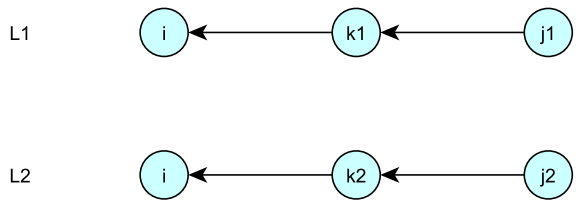

Figura 42: Exemplo do critério de dominância para o caminho regressivo.

6.3 Heurísticas

Esta seção descreve as heurísticas utilizadas para gerar rotas que dêem origem a colunas com custo relativo negativo e, assim, possam melhorar o desempenho do algoritmo de programação dinâmica, resultando em um método de geração de colunas mais eficiente. Foram implementadas heurísticas construtivas que servem para gerar rotas que dão origem às colunas iniciais do problema mestre e também gerar rotas que são utilizadas na programação dinâmica. Além disso, foram implementadas heurísticas de melhoria, a fim de melhorar uma rota existente, gerando uma nova rota com custo relativo negativo menor que a original.

6.3.1 Heurística pra gerar rotas iniciais

A heurística utilizada pra gerar colunas iniciais para o problema mestre e também auxiliar na programação dinâmica foi uma adaptação da heurística de Xu et al. (2003), em que os autores resolvem o PDPTW através de um heurística que possui dois passos: de construção, verificando apenas a precedência e capacidade, e depois de *scheduling*, que verifica as janelas de tempo. O resultado é a rota em que o *scheduling* é possível e com menor custo.

No nosso caso, primeiramente são construídas as rotas com apenas um cliente, isto é, o navio sai do depósito inicial s_0, coleta o pedido $i \in P$, entrega este pedido em $i + n$ e depois termina sua rota no depósito final en_0. Desta maneira, cada navio atende somente a um pedido semelhante à estratégia usada na heurística de economias (*savings*) proposta por Clarke e Wright (1964).

O próximo passo é juntar duas rotas de apenas um cliente, digamos r_1 e r_2. O algoritmo sempre tenta juntar a rota r_1 com alguma que ainda seja composta por apenas um cliente (r_2). Caso não seja possível juntar r_2 à primeira, então r_2 volta para uma lista de rotas com apenas um cliente, para que, depois, a verificação seja feita com uma terceira rota r_3, e assim sucessivamente. Há vários critérios de prioridades para listar todas as rotas com apenas um nó de coleta e entrega, e sendo assim, o algoritmo é executado para todos os critérios de modo a encontrar várias rotas iniciais para o problema.

Ao tentar juntar duas rotas, primeiro é verificado a precedência dos elementos. Depois, calcula-se o custo associado a cada nova rota gerada; e por último, verifica-se as janelas de tempo, capacidade, adjacência, atracação, calado flexível e posicionamento dinâmico. São consideradas apenas as rotas em que a junção foi possível em relação a todos os recursos. Neste momento, temos duas situações: 1) todas as rotas geradas por essa heurística são acrescentadas às colunas iniciais do problema mestre; 2) somente as rotas com custo relativo negativo são utilizadas na programação dinâmica. O Algoritmo 11 mostra como esta heurística foi implementada. Nele, temos que todas as rotas estão em uma lista de rotas R de prioridades e tenta-se juntar as rotas r_1 e r_2. Inicialmente temos que $MelhorEconomia$ é igual a 0, considerando que ainda nenhuma rota foi eliminada. O nó i inicia no depósito inicial (s_0) e varia até o depósito final en_0. O nó j inicia em i e também varia até o depósito final. A cada variação de i e j calcula-se a economia de se eliminar a rota r_2 ao inserir seus nós em r_1 ($Economia$). Então, se $Economia$ for maior que $MelhorEconomia$ e a nova rota for factível, então $MelhorEconomia$ é atualizada. Somente a rota referente a $MelhorEconomia$ é considerada no final do algoritmo.

Algoritmo 11: Heurística construtiva.

```
1  enquanto Existirem rotas na lista R faça
2      Selecione a primeira rota r_1;
3      Selecione a segunda rota r_2;
4      i = s_0;
5      enquanto i < en_0 faça
6          j = i;
7          enquanto j < en_0 faça
8              se Economia > MelhorEconomia então
9                  Checa a factibilidade ao inserir i e j em r_1;
10                 se Rota factível então
11                     MelhorEconomia = Economia;
12             j = next_j (próximo nó de r_1);
13         i = next_i (próximo nó de r_1);
14     Inserir rota correspondente ao MelhorCusto;
```

Para exemplificar, vamos supor que temos duas rotas, cada uma com apenas um nó de coleta e seu respectivo nó de entrega:

$$r_1 = s_0 \rightarrow i \rightarrow i + n \rightarrow en_0$$

$$r_2 = s_0 \rightarrow j \rightarrow j + n \rightarrow en_0$$

O algoritmo tenta juntar estas duas rotas. Primeiramente são verificadas precedência e capacidade. Sendo assim, temos que verificar seis rotas diferentes em relação à precedência:

1. $s_0 \rightarrow i \rightarrow j \rightarrow i + n \rightarrow j + n \rightarrow en_0$;
2. $s_0 \rightarrow j \rightarrow i \rightarrow i + n \rightarrow j + n \rightarrow en_0$;
3. $s_0 \rightarrow i \rightarrow j \rightarrow j + n \rightarrow i + n \rightarrow en_0$;
4. $s_0 \rightarrow j \rightarrow i \rightarrow j + n \rightarrow i + n \rightarrow en_0$;
5. $s_0 \rightarrow i \rightarrow i + n \rightarrow j \rightarrow j + n \rightarrow en_0$;
6. $s_0 \rightarrow j \rightarrow j + n \rightarrow i \rightarrow i + n \rightarrow en_0$.

Vamos supor que apenas as rotas 1, 5 e 6 são factíveis em relação à capacidade. Então, calcula-se o custo destas três rotas e depois verifica-se as janelas de tempo, atracação, calado flexível e posicionamento dinâmico. Vamos supor que apenas as rotas 1 e 5 são factíveis em relação a todos os recursos e que possuem custo relativo negativo. Sendo assim, estas duas rotas são consideradas como rotas novas e que depois, tentarão se juntar a outras rotas com um único par de coleta e entrega.

Generalizando, o algoritmo tenta juntar rotas com uma única coleta e entrega com outras rotas, estas rotas de um único par estão em uma lista de prioridades. Vamos supor então que o algoritmo tente juntar a rota $s_0 \rightarrow k \rightarrow k+n \rightarrow en_0$ à rota $s_0 \rightarrow i \rightarrow j \rightarrow i+n \rightarrow j+n \rightarrow en_0$ (r_1). Então, primeiro é testado a inserção dos nós k e $k+n$ juntos e logo após o nó s_0. Depois, o teste é feito com k depois do nó s_0 e com $k+n$ variando na rota 1. A Figura 43 ilustra os primeiros passos deste exemplo. A primeira parte da figura representa a rota 1. Na segunda parte da figura é feita a tentativa de colocar o par k e $k+n$ antes de i. Na terceira parte da figura o nó k é fixado antes de i e movimenta somente o nó $k+n$, em que a tentativa é colocá-lo antes de j. Na quarta parte da figura o nó k continua antes de i e movimenta-se novamente o nó $k+n$, em que a tentativa é colocá-lo antes de $i+n$. Depois, ambos k e $k+n$ são testados antes do nó j, e assim sucessivamente. Assim, o algoritmo encontrará a melhor rota com o melhor *saving*, correspondente à economia de se eliminar a rota de um único par ao inserir os nós dessa rota em outra já existente.

6.3.2 Heurísticas de melhoria

Esta subseção descreve as heurísticas de melhoria utilizadas para melhorar as rotas já existentes. Depois de geradas todas as rotas pela heurística construtiva, as heurísticas de melhoria são chamadas para melhorar tais rotas. Foram implementadas três tipos de heurísticas de melhoria.

- **Incluir um novo par de clientes a uma rota já existente:**

  Nesta heurística de melhoria, tenta-se incluir um novo par coleta/entrega a uma rota já existente. Todos as rotas com custo relativo negativo geradas por essa heurística são acrescentadas à lista de rotas geradas, ou seja, às colunas do problema mestre.

  Para verificar se um par de coleta e entrega pode ser acrescido à rota existente, temos que considerar adjacência, precedência, janelas de tempo, capacidade, atracação, calado flexível e posicionamento dinâmico. Primeiro, seleciona-se um par

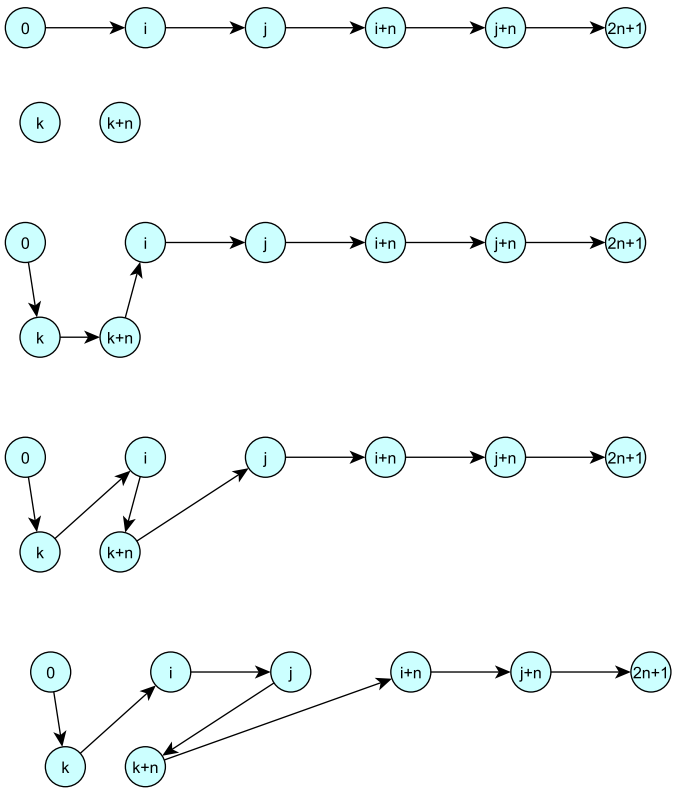

Figura 43: Exemplo de como a heurística para gerar rotas iniciais é executada.

de coleta e entrega. Depois, tenta-se colocar este par na primeira posição da rota, primeiramente juntos, digamos $i, i+n$. Depois, o nó de entrega varia na rota para outras posições, enquanto que o nó de coleta permanece na primeira posição. Quando a rota termina para esta primeira verificação, o par de coleta e entrega é testado na segunda posição da rota, e assim sucessivamente. Depois de todas as tentativas com esse par de coleta e entrega, outro par é testado nesta mesma rota. Temos que garantir que o par testado não esteja na rota já existente.

- **Retirar um par de clientes de uma rota já existente:**

  Nesta heurística tenta-se retirar um par de coleta e entrega de uma rota já existente. Sendo assim, somente as rotas que forem melhoradas (custo relativo mais negativo) serão acrescidas às colunas do problema mestre. Somente as coletas são verificadas, desta maneira, se uma coleta for retirada, sua respectiva entrega também tem de ser retirada da rota. Precisa-se verificar a adjacência dos nós da rota. A precedência, capacidade, janelas de tempo, atracação, calado flexível e posicionamento dinâmico não precisam ser verificadas, já que ao retirar um par de coleta/entrega estas características não serão alteradas.

- **Trocar dois nós consecutivos de uma rota já existente:**

  Nesta heurística tenta-se trocar dois nós adjacentes, ou seja, se temos uma rota com os nós $s-t-u-v$, tentamos trocar os nós t e u para resultar na rota $s-u-t-v$. Apenas as rotas com custo reduzido mais negativo são acrescidas à lista de rotas geradas.

  As janelas de tempo, adjacência, atracação, calado flexível e posicionamento dinâmico precisam ser verificadas em todos os casos. A precedência somente precisa ser verificada no caso em que t é a coleta de u, neste caso a troca não pode ser realizada. Quanto à capacidade, esta deve ser verificada no caso em que t é uma entrega e u é uma coleta (DP). Neste caso, se a troca for realizada pode ser que a capacidade seja violada, pois se s for uma coleta ao se juntar com a demanda de u, pode violar a capacidade do navio.

- **Heurística primal (MIP):**

  O uso desta heurística no método *branch-and-price* foi motivado pelo fato de que a estrutura da formulação de particionamento de conjuntos é adequada para a implementação de heurísticas MIP. Podemos tentar obter uma solução inteira factível combinando algumas colunas do PMR impondo a integralidade das variáveis do pro-

blema mestre, e resolvendo este problema através de um *software* de otimização de propósito geral como, por exemplo, o CPLEX. A fim de garantir que esta heurística seja rápida, limitamos o tempo em que o *solver* é executado para encontrar uma solução inteira factível. A solução obtida pode não ser a ótima para o problema original, mas alguns estudos da literatura mostraram que as soluções obtidas por esta heurística são bons limitantes superiores para o método.

6.4 Estratégias de Ramificação

Diferentes estratégias de ramificação foram implementadas. Tais estratégias são dadas de forma hierárquica: primeiro o método tenta ramificar no número de navios, se não conseguir, o próximo passo é tentar ramificar em um navio específico, se não for possível, então a ramificação é feita nos arcos. Estas três estratégias de ramificação estão detalhadas abaixo:

1. Ramificação no número de navios: em um determinado nó, se a solução do problema mestre for fracionária, então verifica-se a quantidade de navios utilizada. Se esta quantidade for fracionária, então a ramificação é feita no número de navios, isto é, dado $\bar{y}_r$ o valor fracionário de cada variável y_r da solução ótima do problema mestre linear, seja nV o número de navios utilizados nesta solução, então $nV = \sum_{k \in K} \sum_{r \in \Omega_k} \bar{y}_r$. Se nV for fracionário, então a ramificação é feita criando dois nós filhos em que adiciona-se uma das seguintes inequações diretamente ao problema mestre de cada um:

$$\sum_{k \in K} \sum_{r \in \Omega_k} \bar{y}_r \leq \lfloor nV \rfloor ,$$

$$\sum_{k \in K} \sum_{r \in \Omega_k} \bar{y}_r \geq \lceil nV \rceil ;$$

2. Ramificação no veículo: se o valor nV for inteiro, então a ramificação é feita em um navio específico. Seja nK_k o número de vezes que o navio do tipo k é utilizado na solução ótima do problema mestre linear, com $nK_k = \sum_{r \in \Omega_k} \bar{y}_r$. Se algum nK_k for fracionário para $k \in K$, então seleciona-se a variável nK_k mais próxima do valor 0,5 para que a ramificação seja realizada. Em um nó filho será fixado que o navio específico seja utilizado e no outro, será fixado que o navio específico não seja utilizado, ou seja, em cada nó filho, é adicionado uma das seguites inequações ao

problema mestre:

$$\sum_{r \in \Omega_k} \bar{y}_r \leq \lfloor nK_k \rfloor ,$$

$$\sum_{r \in \Omega_k} \bar{y}_r \geq \lceil nK_k \rceil ;$$

3. Ramificação nos arcos: se o valor de nK_k for inteiro para todo $k \in K$, então a ramificação é feita nos arcos da rede. Dada uma solução fracinária do problema mestre $\bar{y}_r$, a solução em termos dos arcos de cada navio k é dada por $\bar{x}_{ijk} = \sum_{r \in \Omega_k} \bar{x}^r_{ijk} \bar{y}_r, \forall k \in K, i,j \in N$, em que $\bar{x}^r_{ijk} = 1$ se e somente se, na rota $r \in \Omega_k$, o navio k viaja diretamente do nó i para o nó j e $\bar{x}^r_{ijk} = 0$, caso contrário. O valor $\bar{x}_{ijk}$ pode ser fracionário devido ao $\bar{y}_r$ fracionário. Sendo assim, a ramificação é feita para algum $k \in K$, cujo valor $\bar{x}_{ijk}$ seja fracionário, em que em um dos nós filhos é imposto $\bar{x}_{ijk} = 0$ e no outro, é imposto $\bar{x}_{ijk} = 1$. Note que neste tipo de ramificação, a imposição dos arcos é feita somente no subproblema k referente ao $\bar{x}_{ijk}$ fracionário, e não mais diretamente no problema mestre.

6.5 Experimentos Computacionais

Apresentamos nesta seção os resultados computacionais para o método *branch-and-price* proposto neste capítulo. Os testes foram realizados com os mesmos exemplares reais fornecidos pela empresa do estudo de caso e analisados nos capítulos anteriores. Cada caso teste foi subdividido em exemplares menores, com o mesmo número de coletas e entregas dos testes com o modelo matemático e com os métodos *branch-and-cut*. O computador utilizado foi o mesmo já mencionado no texto, ou seja, um PC Core *i*7 3.40 GHz, 16 GB de memória RAM, o sistema operacional é outro, agora utilizamos do Linux Ubuntu. O *solver* utilizado na heurística primal (MIP) foi o CPLEX versão 12.4. O tempo computacional limite foi o mesmo utilizado nos outros métodos dos capítulos anteriores, ou seja, de 5 horas de processamento.

6.5.1 Detalhes da Implementação

Nesta subseção, abordamos alguns detalhes importantes da implementação do método *branch-and-price*. Usualmente em trabalhos que envolvem o método *branch-and-price*, o método *simplex* é utilizado para resolver o problema mestre restrito na geração de colunas. Porém, este método pode prejudicar o desempenho da geração de colunas, uma vez que

a oscilação entre os pontos extremos de uma iteração a outra pode resultar em uma convergência mais lenta. Sendo assim, estratégias com soluções não-extremais tem sido aplicadas como, por exemplo, o método primal-dual de pontos interiores (GONDZIO et al., 2013).

O algoritmo *branch-and-price* desta tese foi derivado de um *framework* implementado na linguagem C, em que os autores utilizam da técnica primal-dual de pontos interiores na geração de colunas (MUNARI; GONDZIO, 2013). Os autores desenvolveram um algoritmo *branch-price-and-cut* de pontos interiores para o problema de roteamento de veículos com janelas de tempo. A ramificação é feita antes de se obter a solução ótima do problema mestre restrito. Isto é feito porque o método de pontos interiores encontra uma solução próxima da solução ótima rapidamente, mas necessita de grande esforço computacional para obter a solução ótima. A seguinte estratégia é realizada: a ramificação é feita em duas fases. Na primeira, o objetivo é encontrar uma solução sub-ótima do problema mestre restrito do nó atual com tolerância $\epsilon = 10^{-3}$. Se o nó não for elegível para ser eliminado da árvore *branch-and-bound*, então a ramificação é feita e a segunda fase não é necessária. Caso contrário, a segunda fase é iniciada e o problema mestre restrito é resolvido otimamente. Além disso, simplificações são feitas para que cada subproblema seja resolvido mais rapidamente, isto é, a dominância é verificada em um número limitado de nós, o número de *labels* é limitado e cada *label* é estendido até um certo número de nós (MUNARI; GONDZIO, 2013).

6.5.2 Resultados

As Tabelas 12 e 13 mostram os resultados para o método *branch-and-price* completo, ou seja, com todas as características descritas neste capítulo. Nas colunas 2 a 4 de cada tabela é mostrado o melhor resultado de cada exemplar para o método *branch-and-cut*, apresentando-se o melhor valor da solução (UB), junto com o tempo dado em segundos e com o *gap* dado em porcentagem, extraídos do Capítulo 5. Se mais de um método obteve a melhor solução, então o critério de desempate foi o de melhor tempo, seguido do melhor *gap*. As outras colunas das tabelas mostram os resultados para o método *branch-and-price*. Temos o valor da solução (UB), o tempo total dado em segundos, o *gap* dado em porcentagem, a relaxação linear (RL), o tempo em que o método demorou para encontrar a melhor solução, número de nós, número de colunas, tempo total gasto para resolver o problema mestre (segundos) e o tempo total gasto para gerar as colunas (segundos). Os espaçamentos em branco indicam que o método não encontrou solução factível no tempo

limite estipulado e o símbolo − indica que o método parou por limite de tempo, mas com solução factível.

Para o caso teste 1, o método *branch-and-cut* resolveu otimamente e em pouco tempo computacional exemplares com até 20 pares de coleta e entrega. Porém, para os exemplares $C1N25$ e $C1N30$, o método encontrou solução factível em 5 horas de processamento, mas terminou com *gap* maior que zero. Por outro lado, o método *branch-and-price* resolveu otimamente até o exemplar $C1N50$, ou seja, 50 pares de coleta e entrega, sendo que com até 45 pares de coleta e entrega, os exemplares foram resolvidos em menos de 1 hora de processamento. Para os demais exemplares ($C1N55$ até $C1N80$), o método encontrou solução factível, mas não provou ser a solução ótima, terminando com *gap* positivo. Um importante resultado é que os *gaps* não ultrapassaram os 4% para este caso, ou seja, as soluções estão relativamente próximas da solução ótima. Cabe dizer que para a maioria desses exemplares, o método encontrou a melhor solução inteira em pouco tempo computacional, exceto para o $C1N75$, a qual demorou mais de 14000 segundos.

Para o caso teste 2, os resultados são um pouco diferentes. O método *branch-and-cut* resolve otimamente exemplares com até 20 pares de coleta e entrega, sendo que o exemplar $C2N20$ é resolvido otimamente em tempo computacional menor, comparado ao método *branch-and-price* (445 do primeiro contra 3190 do segundo). Para o $C2N25$, ambos os métodos encontram a mesma solução inteira, e nenhum método prova sua otimalidade. Para os demais exemplares $C2N30$ até $C2N80$, somente o método *branch-and-price* encontra solução factível em 5 horas de processamento. O *gap* ao final de 5 horas de processamento para este caso não ultrapassa os 3%, com maior *gap* de 2,79% para o $C2N50$. Sendo assim, para ambos os casos teste, o método *branch-and-price* encontra solução factível para todos os exemplares.

Em relação ao número de nós e o número de colunas, os exemplares do Caso 1 possuem menor número comparados aos exemplares do Caso 2. Por exemplo, o exemplar do Caso 1 com o maior número de nós é o $C1N55$, com 124 nós, comparado ao $C2N25$, com 5172 nós. Além disso, o exemplar com maior número de colunas do Caso 1 é o $C1N70$, com 9461 colunas, comparado ao $C2N20$, com 19727 colunas. Para todos os exemplares de ambos os casos, em geral o tempo para resolver o problema mestre é muito menor que o tempo total para gerar as colunas, o que não comprometeu o valor da relaxação linear. Este fato se deve ao tempo gasto na programação dinâmica e também nas heurísticas construtivas e de melhoria.

Outro teste realizado foi com o método *branch-and-price* sem as heurísticas do subpro-

Tabela 12: Resultados computacionais para o Caso 1.

Exem.	*Branch-and-Cut*			*Branch-and-Price*								
	UB	Tempo (s)	*Gap* (%)	UB	Tempo (s)	*Gap* (%)	RL	Tempo melhor sol.	# Nós	# Col	Tempo Mestre (s)	Tempo Colunas (s)
$C1N10$	1677,84	6,49	0	1677,84	7	0	1515,97	0	7	109	0,06	6,28
$C1N15$	2311,98	14,99	0	2311,98	141	0	2145,21	1	115	266	1,49	137,86
$C1N20$	2748,36	24,19	0	2748,36	3	0	2748,36	1	1	333	0,02	2,94
$C1N25$	3522,90	–	16,19	3522,90	6	0	3522,90	2	1	561	0,04	6,00
$C1N30$	4691,25	–	46,40	4596,99	75	0	4443,90	45	13	838	0,56	73,62
$C1N35$				4814,04	184	0	4645,74	38	5	1090	0,45	183,02
$C1N40$				5477,45	663	0	5476,20	102	5	1418	0,55	662,43
$C1N45$				5924,52	298	0	5924,52	150	1	1668	0,22	296,65
$C1N50$				6409,89	5386	0	6409,87	185	23	2179	3,82	5381,40
$C1N55$				6804,11	–	0,15	6793,59	4007	124	3937	14,45	17980,37
$C1N60$				7398,40	–	0,37	7371,02	1188	65	5778	13,16	17981,99
$C1N65$				7862,95	–	0,55	7819,36	4135	41	7643	11,69	17982,06
$C1N70$				8558,70	–	2,87	8312,67	4773	30	9461	11,85	17981,71
$C1N75$				8810,63	–	1,56	8673,31	14384	22	8074	9,01	17983,80
$C1N80$				9349,68	–	1,07	9249,36	6615	23	7103	9,56	17983,80
Média	–	14403,04	70,83	–	7650,86	0,44	–	2375,07	31,73	3363,87	5,13	7642,93

Tabela 13: Resultados computacionais para o Caso 2.

Exem.	*Branch-and-Cut*			*Branch-and-Price*								
	UB	Tempo (s)	*Gap* (%)	UB	Tempo (s)	*Gap* (%)	RL	Tempo melhor sol.	# Nós	# Col	Tempo Mestre (s)	Tempo Colunas (s)
$C2N10$	1402,63	16,34	0	1402,63	3	0	1402,62	0	1	239	0,01	3,03
$C2N15$	1708,51	24,78	0	1708,50	7	0	1659,37	1	3	528	0,07	6,61
$C2N20$	2452,90	445,33	0	2452,89	3190	0	2403,85	2	861	19727	24,04	3158,44
$C2N25$	3240,60	–	0,93	3240,60	–	1,91	3178,79	3	5172	83746	251,66	17646,48
$C2N30$				3810,71	–	1,69	3746,44	13	1250	39765	59,18	17905,62
$C2N35$				4308,53	–	1,34	4250,83	5369	696	9906	35,82	17940,35
$C2N40$				5333,47	–	1,43	5257,22	1077	557	12807	45,91	17930,54
$C2N45$				5520,86	–	1,43	5442,01	224	305	5121	21,39	17963,58
$C2N50$				6294,72	–	2,00	6169,04	4006	136	5613	19,07	17971,87
$C2N55$				6484,64	–	1,14	6410,45	382	95	6138	12,11	17979,55
$C2N60$				7066,11	–	0,82	7008,03	11627	60	9223	11,49	17980,39
$C2N65$				7518,09	–	1,38	7414,08	1160	55	6178	8,69	17983,11
$C2N70$				8034,23	–	0,75	7973,83	6084	88	8518	21,92	17934,09
$C2N75$				8471,50	–	1,42	8351,15	17613	122	18851	55,64	17894,13
$C2N80$				8603,09	–	1,72	8455,13	16464	21	9121	8,65	17964,75
Média	–	14432,43	73,39	–	14613,33	1,14	–	4268,33	628,13	15698,73	38,38	14550,84

blema das Seções 6.3.1 e 6.3.2, ou seja, sem as heurísticas iniciais e de melhoria descritas neste capítulo. O intuito destes testes foi analisar o quão importante era o uso destas heurísticas no método. As Tabelas 14 e 15 descrevem estes resultados A primeira coluna mostra o nome do exemplar, a segunda o valor da solução (UB), a terceira coluna mostra a razão entre o valor UB do resultado sem as heurísticas e o valor UB do resultado com as heurísticas (Tabelas 12 e 13). A quarta coluna mostra o tempo total dado em segundos, seguido do *gap* dado em porcentagem, do tempo para encontrar a melhor solução (segundos), do número de nós, número de colunas, tempo gasto no problema mestre dado em segundos e o tempo gasto para gerar as colunas (segundos).

Para o caso de teste 1, os resultados foram parecidos até o exemplar $C1N45$, ou seja, o método com as heurísticas e sem as heurísticas resolveu otimamente em pouco tempo computacional os exemplares com 10 a 45 pares de coleta e entrega. Para o $C1N50$, o método sem as heurísticas resolveu otimamente em menor tempo computacional, 1693 comparado ao método com as heurísticas, que resolveu em 5386 segundos. Para $C1N55$ até $C1N65$, ambos os métodos encontraram a mesma solução no tempo estipulado de 5 horas. Para $C1N70$ e $C1N80$, o método sem as heurísticas encontrou uma solução melhor e para $C1N75$, o método com as heurísticas encontrou uma solução melhor, isto pode ser visto na coluna "razão" da tabela. Em relação às médias do tempo total gasto, *gap*, tempo para encontrar a melhor solução, número de colunas, tempo gasto no problema mestre e tempo gasto para gerar as colunas, o método sem as heurísticas obteve resultado melhor. O método completo (com as heurísticas - Tabela 12) obteve melhor média somente no número de nós.

Para o caso de teste 2, os resultados foram diferentes, uma vez que para vários exemplares o método sem as heurísticas obteve resultados diferentes. Para os exemplares $C2N60$, $C2N70$ e $C2N75$, o método com as heurísticas obteve melhores resultados. Para $C2N35$, $C2N45$, $C2N65$ e $C2N80$, o método sem as heurísticas obteve soluções melhores no limite de 5 horas de processamento (coluna "razão" da tabela). Para os demais exemplares, ambos os métodos obtiveram resultados iguais ou semelhantes em relação ao tempo computacional. Quanto as médias, o Caso 2 obteve resultados diferentes em relação ao Caso 1. O método sem as heurísticas obteve média superior ao método com as heurísticas somente em relação ao *gap* e também ao tempo gasto para encontrar a melhor solução. Em relação às outras colunas da tabela, o método com as heurísticas obteve melhor média.

Estes resultados mostraram que, para alguns exemplares, o método sem as heurísticas

Tabela 14: Resultados computacionais para o Caso 1.

Exem.	*Branch-and-Price*								
	UB	Razão	Tempo (s)	*Gap* (%)	Tempo melhor sol.	# Nós	# Col	Tempo Mestre	Tempo Colunas
$C1N10$	1677,84	1,000	6	0	1	7	111	0,06	5,72
$C1N15$	2311,98	1,000	140	0	1	163	273	2,04	135,94
$C1N20$	2748,36	1,000	1	0	0	1	336	0,02	1,62
$C1N25$	3522,9	1,000	3	0	1	1	551	0,04	2,61
$C1N30$	4596,99	1,000	54	0	3	17	811	0,59	52,89
$C1N35$	4814,04	1,000	159	0	36	5	886	0,35	158,24
$C1N40$	5477,45	1,000	807	0	101	5	1164	0,60	806,21
$C1N45$	5924,52	1,000	320	0	180	1	1287	0,19	320,37
$C1N50$	6409,89	1,000	1693	0	243	7	1646	1,12	1691,61
$C1N55$	6804,11	1,000	–	0,15	316	168	4779	14,94	17979,06
$C1N60$	7398,40	1,000	–	0,37	928	56	8017	12,54	17984,56
$C1N65$	7862,95	1,000	–	0,55	1316	31	7962	8,54	17987,16
$C1N70$	8531,51	0,997	–	2,57	8458	26	9103	7,36	17988,16
$C1N75$	8816,64	1,001	–	1,63	3464	13	6810	6,47	17991,36
$C1N80$	9328,03	0,998	–	0,84	3166	12	6242	10,00	17988,11
Média	–	–	7412,20	0,41	1214,27	34,20	3331,87	4,32	7406,24

Tabela 15: Resultados computacionais para o Caso 2.

Exem.	*Branch-and-Price*								
	UB	Razão	Tempo (s)	*Gap* (%)	Tempo melhor sol.	# Nós	# Col	Tempo Mestre	Tempo Colunas
$C2N10$	1402,62	1,000	3	0	1	1	272	0,02	2,43
$C2N15$	1708,50	1,000	4	0	1	3	647	0,08	3,72
$C2N20$	2452,89	1,000	3825	0	2	1863	22164	46,29	3761,58
$C2N25$	3240,60	1,000	–	1,91	2	7371	88526	309,09	17557,26
$C2N30$	3810,71	1,000	–	1,69	15	1931	59410	67,92	17876,94
$C2N35$	4301,12	0,998	–	1,17	86	800	18251	27,00	17948,62
$C2N40$	5333,47	1,000	–	1,43	730	281	17074	25,51	17965,05
$C2N45$	5518,26	1,000	–	1,38	1089	204	9796	15,00	17976,47
$C2N50$	6294,72	1,000	–	2,00	8655	82	9067	9,93	17985,79
$C2N55$	6484,64	1,000	–	1,14	515	103	4535	7,10	17989,09
$C2N60$	7087,53	1,003	–	1,12	537	67	7007	8,50	17988,28
$C2N65$	7514,26	0,999	–	1,33	6375	44	5110	5,56	17990,89
$C2N70$	8045,02	1,001	–	0,88	14632	28	8803	8,56	17985,88
$C2N75$	8479,09	1,001	–	1,51	17555	209	20454	48,57	17931,34
$C2N80$	8600,87	1,000	–	1,69	4064	28	4751	5,69	17987,27
Média	–	–	14655,47	1,15	3617,27	867,67	18391,13	38,99	13596,71

foi melhor que o método com as heurísticas. Isto pode ter ocorrido devido ao tempo gasto com as heurísticas em cada iteração, uma vez que estas são chamadas para cada navio na resolução do subproblema. Algumas melhorias poderiam ainda ser feitas para que este tempo computacional seja menor e assim, fazer com que os resultados do método com as heurísticas se sobreponha em todos os exemplares ao método sem as heurísticas.

O objetivo da próxima tabela foi comparar as diferentes estratégias de ramificação aplicadas. Para esta análise, somente os exemplares do Caso 1 foram testados. Na Tabela 16 são mostrados os resultados para o Caso 1 com o código completo (dados da Tabela 12), resultados com a ramificação somente nos arcos (Ramificação 1), ramificação no número de navios (Ramificação 2) e ramificação em um navio específico (Ramificação 3). Note que a ramificação nos arcos é a única que garante cortar todas as soluções com x fracionário, então nas Ramificações 2 e 3 a Ramificação 1 também está presente, caso se tenha a soma de $\bar{y}$ (variável do problema mestre) inteiro para o número de navios e também para um navio específico, a ramificação é feita nos arcos. Em cada uma das situações é mostrado o valor da solução para cada exemplar (UB), o tempo em segundos e também o *gap* em porcentagem. Os melhores resultados para cada exemplar estão destacados em negrito na tabela.

Os resultados da Tabela 16 mostram que os resultados com o código completo, ou seja, com as três estratégias de ramificação juntas, obtém os melhores resultados para o Caso 1, uma vez que para todos os exemplares deste caso, exceto para o $C1N80$, obtém a melhor solução no tempo estipulado e a média do tempo gasto é menor (7650,86). Embora a média dos *gaps* não tenha sido a melhor a diferença não é grande uma vez que a média para o código completo é de 0,44 e a média dos *gaps* para a Ramificação 1 é de 0,43. A abordagem com todas as ramificações juntamente com a Ramificação 2 (número total de navios) foram os únicos métodos capazes de encontrar e provar a solução ótima em menos de 5 horas para os exemplares $C1N10$ até $C1N50$. Portanto, uma boa estratégia é considerar as três estratégias de ramificação, isto é, no número total de navios, em um navio específico e também nos arcos.

As próximas análises foram feitas de acordo com as Tabelas 3, 4 e 5 para analisar porquê o Caso de teste 1 obtém melhores resultados, ou seja, tentar identificar quais características de cada exemplar que resultam no método resolver um maior número de exemplares para o Caso 1 do que para o Caso 2.

Acredita-se que um dos principais motivos pelo o qual o método *branch-and-price* resolve mais exemplares para o Caso 1 (do que o Caso 2) seja o número de navios (Tabela

Tabela 16: Diferentes estratégias de ramificação - Caso 1.

Exem.	Código Completo			Ramificação 1			Ramificação 2			Ramificação 3		
	UB	Tempo (s)	*Gap* (%)	UB	Tempo (s)	*Gap* (%)	UB	Tempo (s)	*Gap* (%)	UB	Tempo (s)	*Gap* (%)
$C1N10$	**1677,84**	7	**0**	**1677,84**	**6**	**0**	**1677,84**	8	**0**	**1677,84**	1433	**0**
$C1N15$	**2311,98**	141	**0**	**2311,98**	**7**	**0**	**2311,98**	9	**0**	**2311,98**	–	7,21
$C1N20$	**2748,36**	**3**	**0**	**2748,36**	**3**	**0**	**2748,36**	**3**	**0**	**2748,36**	4	**0**
$C1N25$	**3522,90**	**6**	**0**	**3522,90**	**6**	**0**	**3522,90**	**6**	**0**	**3522,90**	**6**	**0**
$C1N30$	**4596,99**	75	**0**	**4596,99**	**42**	**0**	**4596,99**	74	**0**	**4596,99**	84	**0**
$C1N35$	**4814,04**	184	**0**	**4814,04**	**183**	**0**	**4814,04**	184	**0**	**4814,04**	187	**0**
$C1N40$	**5477,45**	663	**0**	**5477,45**	–	0,02	**5477,45**	3786	**0**	**5477,45**	**432**	**0**
$C1N45$	**5924,52**	**298**	**0**	**5924,52**	**298**	**0**	**5924,52**	**298**	**0**	**5924,52**	306	**0**
$C1N50$	**6409,89**	5386	**0**	**6409,89**	5372	**0**	**6409,89**	**5372**	**0**	**6409,89**	5479	**0**
$C1N55$	**6804,11**	–	**0,15**	6804,62	–	0,16	**6804,11**	–	**0,15**	**6804,11**	–	**0,15**
$C1N60$	**7398,40**	–	**0,37**	**7398,40**	–	**0,37**	**7398,40**	–	**0,37**	**7398,40**	–	**0,37**
$C1N65$	**7862,95**	–	**0,55**	**7862,95**	–	**0,55**	**7862,95**	–	**0,44**	**7862,95**	–	**0,55**
$C1N70$	**8558,70**	–	**2,87**	8559,42	–	2,88	8568,27	–	2,98	8594,46	–	3,28
$C1N75$	**8810,63**	–	**1,56**	8818,37	–	1,64	**8810,63**	–	**1,56**	8821,40	–	1,68
$C1N80$	9349,68	–	1,07	**9323,14**	–	**0,79**	9335,09	–	0,92	9349,68	–	1,07
Média	–	**7650,86**	0,44	–	8794,46	**0,43**	–	7849,33	0,44	–	8928,73	0,95

3). No primeiro caso, são 25 navios quando comparado ao segundo caso, que são 31 navios. Um maior número de navios faz com que o método *branch-and-cut* tenha mais restrições e variáveis e, quanto ao método *branch-and-price*, faz com que mais subproblemas tenham de ser resolvidos, uma vez que resolve-se um subproblema para cada navio em cada nó da árvore *branch-and-bound*. Outro motivo pode estar relacionados às janelas de tempo. Analisando as Tabelas 4 e 5, nota-se que as janelas de tempo do Caso 2 são mais abertas, comparadas as do Caso 1. Janelas de tempo mais apertadas em geral fazem com que o método *branch-and-price* seja mais eficaz, uma vez que não são muitas as colunas factíveis encontradas.

Do ponto de vista prático, os resultados com o método *branch-and-price* são aceitáveis, uma vez que resolve otimamente em pouco tempo computacional exemplares com até 50 pares de coleta e entrega para o Caso 1 e 20 pares de coleta e entrega para o Caso 2. Além disso, encontra solução factível para até 80 pares de coleta e entrega para ambos os casos. Embora os tempos computacionais sejam elevados para vários exemplares (18000 de processamento), na prática este tempo limite poderia ser menor, dependendo da tolerância do usuário, o que mostra as Tabelas 17 e 18. O tempo limite os resultados mostrados nestas tabelas são de 1800 segundos, ou seja, em meia hora o método é capaz de encontrar boas soluções para todos os exemplares com média dos *gaps* de 0,54 para o Caso 1 e 1,31 para o Caso 2. Além disso, a coluna "razão" mostra os resultados dos limitantes superiores (colunas "UB" das Tabelas 12 e 13) em relação aos limitantes superiores dados em meia hora, isto é, podemos observar que as soluções são próximas, e sendo assim, os resultados são aceitáveis na prática. Uma vez encontrada uma solução factível com *gap* aceitável, cabe aos operadores da empresa decidir entre adotar esta solução, ou permitir que o método seja executado por maior tempo na expectativa de encontrar uma solução melhor.

Portanto, o método *branch-and-price* resolve um maior número de exemplares comparado aos outros métodos aqui explorados (*branch-and-cut* e modelo matemático resolvido pelo *solver* CPLEX). Conforme mencionado, para os exemplares do Caso 1, o método *branch-and-price* obteve soluções ótimas com até 50 pares de coleta e entrega. Para o Caso 2, o método foi capaz de obter soluções ótimas de exemplares com até 20 pares de coleta e entrega. Um resultado importante é que o método encontra solução factível para todos os exemplares testados, ou seja, com até 80 pares de coleta e entrega para ambos os casos de teste reais. Desta maneira, os objetivos propostos nesta tese foram alcançados de forma satisfatória, uma vez que o método *branch-and-price* é capaz de resolver exemplares de tamanho moderado em pouco tempo computacional e assim, tem potencial para

Tabela 17: Resultados computacionais para o Caso 1.

Exem.	*Branch-and-Price*								
	UB	Razão	Tempo (s)	*Gap* (%)	Tempo melhor sol.	# Nós	# Col	Tempo Mestre	Tempo Colunas
$C1N10$	1677,84	1,000	6	0	0	7	109	0,06	6,34
$C1N15$	2311,98	1,000	139	0	1	115	266	1,52	136,59
$C1N20$	2748,36	1,000	3	0	1	1	333	0,02	2,87
$C1N25$	3522,90	1,000	7	0	2	1	561	0,04	6,00
$C1N30$	4596,99	1,000	74	0	45	13	838	0,55	73,94
$C1N35$	4814,04	1,000	184	0	39	5	1090	0,41	182,87
$C1N40$	5477,45	1,000	666	0	102	5	1418	0,54	664,29
$C1N45$	5924,52	1,000	196	0	148	1	1668	0,24	295,79
$C1N50$	6409,89	1,000	–	0,00	186	7	2037	1,38	1798,12
$C1N55$	6804,62	1,000	–	0,16	267	8	3229	1,33	1797,3
$C1N60$	7398,40	1,000	–	0,37	1193	5	3935	1,84	1796,57
$C1N65$	7888,89	1,003	–	0,88	1593	5	3350	1,38	1797,04
$C1N70$	8607,31	1,006	–	3,42	841	4	4147	1,18	1795,77
$C1N75$	8841,63	1,004	–	1,90	1127	1	5468	1,05	1795,79
$C1N80$	9375,73	1,003	–	1,35	1800	1	5393	1,31	1794,91
Média	–	–	925,00	0,54	1214,27	11,93	2256,13	0,86	929,61

Tabela 18: Resultados computacionais para o Caso 2.

Exem.	*Branch-and-Price*								
$C2N10$	1402,62	1,000	3	0	0	1	239	0,01	2,74
$C2N15$	1708,50	1,000	6	0	1	3	528	0,07	5,88
$C2N20$	2452,89	1,000	–	2,00	2	533	9910	13,40	1780,36
$C2N25$	3240,60	1,000	–	1,91	4	531	5923	30,38	1758,33
$C2N30$	3810,71	1,000	–	1,69	13	117	4802	5,48	1790,87
$C2N35$	4313,22	1,001	–	1,45	42	64	1928	2,90	1794,05
$C2N40$	5333,47	1,000	–	1,43	1078	30	3473	3,24	1794,47
$C2N45$	5520,86	1,000	–	1,43	224	25	2641	2,08	1796,62
$C2N50$	6298,55	1,001	–	2,06	369	11	3215	1,50	1796,59
$C2N55$	6484,64	1,000	–	1,14	382	7	3347	1,06	1796,97
$C2N60$	7070,65	1,001	–	0,89	1666	5	4172	0,85	1796,18
$C2N65$	7518,09	1,000	–	1,38	1163	3	4632	0,77	1796,35
$C2N70$	8038,70	1,001	–	0,81	913	5	6277	1,73	1793,85
$C2N75$	8483,14	1,001	–	1,56	405	13	6947	5,28	1784,63
$C2N80$	8623,56	1,002	–	1,95	1586	1	7167	1,08	1790,13
Média	–	–	1560,60	1,31	523,20	89,93	4346,73	4,66	1551,87

resolver problemas reais de empresas petrolíferas.

7 Conclusão

O objetivo deste trabalho foi estudar o problema do transporte de óleo cru de plataformas *offshore* para os terminais localizados na costa brasileira e propor modelos e métodos para sua solução. Dessa forma, foi realizado um estudo de caso com uma empresa brasileira que realiza esta operação de extração e transporte do óleo cru e foram desenvolvidos métodos de solução do tipo *branch-and-cut* e *branch-and-price* especializados para esse problema. Um modelo de programação inteira mista foi proposto, baseado no problema de coleta e entrega com janelas de tempo (ROPKE; CORDEAU, 2009), e considerando várias restrições específicas desse problema, por exemplo, múltiplos depósitos, atracação dos navios nos terminais, calado flexível, dispositivos de posicionamento dinâmico, dentre outras. Não foram encontrados trabalhos na literatura nesta linha de pesquisa que englobem todas estas características específicas desse problema de roteamento na indústria petrolífera. Uma contribuição importante dessa tese é o estudo e a proposta de abordagens de solução para otimização deste problema na prática, com foco em métodos exatos, para ajudar a preencher essa lacuna na literatura.

Este problema foi inicialmente modelado na linguagem OPL, da IBM CPLEX e resolvido pelo *solver* CPLEX versão 12.5.1. Para o desenvolvimento dos testes computacionais a empresa desse estudo disponibilizou exemplares reais referentes a diferentes meses de operação, denominados casos de teste 1 e 2, respectivamente. Os resultados mostraram que para ambos os casos o modelo obteve soluções similares: para o Caso 1, o modelo resolveu otimamente em tempo computacional razoável exemplares com até 20 pares de coleta e entrega. A partir de 25 pares de coleta e entrega, o *solver* atingiu o limite de tempo de 5 horas de processamento com *gap* de otimalidade positivo. Após 25 pares de coleta e entrega, não foram encontradas soluções factíveis (inteiras) em 5 horas de processamento. Para o Caso 2, o modelo resolveu otimamente exemplares com até 15 pares de coleta e entrega. Para 20 e 25 pares, o modelo encontrou soluções factíveis, mas terminou com *gap* positivo após o tempo limite de processamento. Acima de 25 pares de coleta e entrega, o modelo matemático não encontrou soluções factíveis no tempo estipulado.

Sendo assim, o modelo matemático teve dificuldades em resolver exemplares de tamanhos mais realistas, e alguns fatores que podem ter acarretado isto foi o elevado número de variáveis binárias, a relaxação linear relativamente fraca e a simetria do problema. Isto motivou o desenvolvimento de métodos de solução do tipo *branch-and-cut* e *branch-and-price* para resolver otimamente o problema de coleta e entrega com janelas de tempo na indústria petrolífera.

Foram propostos cinco métodos *branch-and-cut*, alguns baseados em modelos com variáveis de 2-índices e outros baseados em modelos com variáveis de 3-índices. Os modelos baseados em variáveis de 2-índices consideram frota homogênea e, sendo assim, um artifício nos dados de entrada e no grafo do modelo permitiu considerar a heterogeneidade dos navios em relação à capacidade, juntamente com os múltiplos depósitos. Vários conjuntos de desigualdades válidas foram implementados, dentre eles: caminhos infactíveis para as limitações de atracação, calado flexível e posicionamento dinâmico, precedência, restrição de capacidade, eliminação de sub-rota, restrição de ordem generalizada, caminhos infactíveis para janelas de tempo e restrições de alcance. Os testes computacionais realizados para os métodos *branch-and-cut* foram feitos com o mesmo conjunto de dados reais. Os resultados mostraram que os métodos baseados em modelos com variáveis de 3-índices, Métodos 4 e 5, foram os que apresentaram os melhores resultados, e tiveram desempenhos superiores ao *solver* CPLEX para resolver o modelo matemático. Além disso, os cortes que tiveram maior impacto no valor do limitante inferior foram os cortes do tipo alcance. Entretanto, os métodos *branch-and-cut* tiveram algumas limitações em resolver exemplares com mais de 30 pares de coleta e entrega. Este fato nos motivou a também desenvolver outro tipo de método exato específico para o problema de coleta e entrega na indústria petrolífera, baseado no método *branch-and-price*.

Foi desenvolvido um método *branch-and-price* específico para o problema, baseado em uma formulação por particionamento de conjuntos com frota heterogênea. Essa formulação possui um número bastante reduzido de restrições em relação às outras formulações, ao custo de um número extremamente grande de variáveis, o que requer o uso da técnica de geração de colunas. Os testes computacionais foram realizados com o mesmo conjunto de dados reais fornecidos pela empresa. Os resultados mostraram que o método *branch-and-price* tem desempenho bem superior aos demais métodos *branch-and-cut* para tratar esse problema, sendo capaz de resolver exemplares com até 50 pares de coleta e entrega para o Caso 1, e com até 20 pares de coleta e entrega para o Caso 2. Além disso, encontra soluções factíveis para todos os exemplares, ou seja, para exemplares com até 80 pares de coleta e entrega em ambos os casos. Este método obteve os melhores resultados para o

conjunto de dados reais, uma vez que resolveu exemplares maiores e encontrou soluções factíveis para todos os exemplares de teste. Os resultados com o método *branch-and-price* são aceitáveis na prática, embora alguns tempos computacionais sejam elevados (18000 segundos), o método encontra solução factível dos exemplares em pouco tempo computacional, e caberia aos operadores da empresa decidir entre utilizar uma solução factível encontrada em pouco tempo computacional, ou executar o método por mais tempo para encontrar uma solução de melhor qualidade.

Conforme discutido no Capítulo 1, os objetivos propostos nesta tese foram alcançados, uma vez que um estudo de caso foi realizado com uma empresa petrolífera e modelos e métodos de solução exatos foram propostos para resolver tal problema. Os resultados mostraram que os métodos têm potencial para resolver problemas em situações reais, notadamente o método *branch-and-price*, que foi o mais bem sucedido.

Ainda é possível fazer várias melhorias nos métodos aqui apresentados. Nos métodos *branch-and-cut*, pode-se incluir e analisar novos cortes (do tipo garfo, outros cortes de capacidade, eliminação de sub-rota, outros cortes de caminhos infactíveis etc). O corte do tipo garfo obteve resultados muito bons na melhoria dos limitantes inferiores para problemas de coleta e entrega na literatura (ROPKE et al., 2007). Pode-se testar outras possibilidades, como o uso de *strong branching* (ACHTERBERG et al., 2005), heurísticas iniciais para gerar boas soluções iniciais e considerar como um limitante superior, explorar simplificações do modelo, dentre outras. Outros tipos de pré-processamento podem ser incluídos para melhorar o desempenho dos métodos, tais como apertar os limitantes das variáveis de tempo e acúmulo de carga (CORDEAU, 2006). Portanto, acredita-se que os resultados podem ainda ser melhorados com a aplicação dessas ideias para tornar os métodos *branch-and-cut* mais competitivos com o método *branch-and-price*, e aumentar seu potencial para resolver problemas realistas em tempo computacional aceitável.

Quanto ao método *branch-and-price*, pode-se melhorar as heurísticas implementadas: inicial e de melhorias, para que o método seja mais eficaz, pode-se também testar outras formas de ramificação e implementar um pré-processamento eficaz. Além disso, uma importante pesquisa futura seria unir os métodos *branch-and-cut* e *branch-and-price* aqui apresentados, ou seja, incluir as desigualdades válidas no *branch-and-price* para melhorar os limitantes inferiores e fazer com que o método resultante (*branch-cut-and-price*) resolva um maior número de exemplares em tempos computacionais menores. Além disso, no método *branch-cut-and-price* pode-se também considerar eliminar um ou mais recursos da programação dinâmica (por exemplo, capacidade, janelas de tempo, dentre outros) e

incluir cortes para que a solução seja factível de acordo com tais recursos não considerados.

Quanto ao modelo matemático aqui proposto, pode-se incluir restrições práticas adicionais como, por exemplo, atracação nos berços dos terminais, considerando de maneira mais detalhada o número de berços de cada terminal e suas limitações para atracação de cada tipo de navio, a compartimentalização dos navios e a consideração de produtos incompatíveis nestes compartimentos do navio, dentre outras. O modelo matemático também pode ser estendido para tratar de particionamento da carga (*splitting delivery*) e situações em que a quantidade carregada não precisa estar definida *a priori*, mas ser uma decisão do problema, como no problema de roteamento com controle de estoques (*inventory routing problem*). Além disso, o Modelo 3 considerado no Método *branch-and-cut* 3, é uma nova estratégia para problemas de coleta e entrega com frota heterogênea, constituindo mais uma contribuição desta tese. Este modelo também pode ser útil para outros problemas de roteamento e esta pode ser uma pesquisa futura promissora, ou seja, o estudo de outras variantes com frota heterogênea que poderiam ser modeladas apropriadamente de acordo com o modelo matemático aqui proposto. Desta maneira, os métodos *branch-and-cut* e *branch-and-price* também poderiam ser adaptados para estas novas versões do modelo matemático, ou seja, incluir nestes métodos outras características da indústria petrolífera ou outras, com novas estratégias de modelagem.

Uma linha de pesquisa futura seria explorar métodos híbridos, em que heurísticas e meta-heurísticas são utilizadas no método *branch-and-price* para gerar e melhorar as rotas para o problema mestre. Métodos híbridos tem-se mostrado bastante promissores para a resolução de problemas de roteamento de veículos e suas variantes, conforme observado mais recentemente na literatura (SUBRAMANIAN et al., 2013).

Uma importante pesquisa futura é a validação na prática de empresas petrolíferas das abordagens aqui propostas. Para tanto, estudos de caso poderiam ser desenvolvidos especialmente com esse objetivo, utilizando diversos exemplares reais, e serem resolvidos pelos métodos propostos nesta tese. Uma comparação e análise detalhada das soluções obtidas pelos métodos com as soluções utilizadas na prática das empresas deveriam ser realizadas, para se verificar melhor a adequação e efetividade dessas abordagens em situações reais. Além da qualidade das soluções, seria interessante comparar o tempo total gasto para obter tais soluções e o *trade-off* entre o tempo e a qualidade das mesmas.

Referências Bibliográficas

ACHTERBERG, T.; KOCH, T.; MARTIN, A. Branching rules revisited. *Operations Research Letters*, v. 33, n. 1, p. 42 – 54, 2005.

AGRA, A.; CHRISTIANSEN, M.; DELGADO, A.; SIMONETTI, L. Hybrid heuristics for a short sea inventory routing problem. *European Journal of Operational Research*, v. 236, n. 3, p. 924 – 935, 2014. Vehicle Routing and Distribution Logistics.

AIZEMBERG, L.; KRAMER, H. H.; PESSOA, A. A.; UCHOA, E. Formulations for a problem of petroleum transportation. *European Journal of Operational Research*, v. 237, n. 1, p. 82 – 90, 2014.

ANDERSSON, H.; HOFF, A.; CHRISTIANSEN, M.; HASLE, G.; LØKKETANGEN, A. Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, v. 37, p. 1515–1536, 2010.

APPELGREN, L. H. A column generation algorithm for a ship scheduling problem. *Transportation Science*, v. 3, p. 53–68, 1969.

APPELGREN, L. H. Integer programming methods for a vessel scheduling problem. *Transportation Science*, v. 5, p. 64–78, 1971.

APPLEGATE, R.; BIXBY, R.; CHVÁTAL; COOK, W. Solving traveling salesman problems. In: . Ann Arbor, MI: 15th International Symposium on Mathematical Programming, 1994.

ARCHETTI, C.; BIANCHESSI, N.; SPERANZA, M. G. A column generation approach for the split delivery vehicle routing problem. *Networks*, v. 58, n. 4, p. 241–254, 2011.

ARCHETTI, C.; SAVELSBERGH, M. W. P.; SPERANZA, G. To split or not to split: That is the question. *Science Direct*, v. 44, p. 114–128, 2008.

ARCHETTI, C.; SPERANZA, M. G. Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, v. 19, n. 1-2, p. 3–22, 2012.

ASCHEUER, N.; FISCHETTI, M.; GRÖTSCHEL, M. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, v. 90, n. 3, p. 475–506, 2001.

ASCHEUER, N.; JÜNGER, M.; REINELT, G. A branch and cut algorithm for the asymetric traveling salesman prolem with precedence constraints. *Computational Optimization and Applications*, v. 17, p. 61–84, 2000.

AUGERAT, P.; BELENGUER, J.; BENAVENT, E.; CORBERÁN, A.; NADDEF, D. Separating capacity constraints in the CVRP using tabu search. *European Journal of Operational Research*, v. 106, n. 23, p. 546–557, 1998.

AUGERAT, P.; BELENGUER, J. M.; BENAVENT, E.; CORBERÁN NADDEF, D.; RINALDI, G. Computational results with a branch and cut code for the capacitated vehicle routing problem. *Technical Report RR 949-M*, Université Joseph Fourier, Grenoble, França, 1995.

BALAS, E.; CERIA, S.; CORNUÉJOLS, G.; NATRAJ, N. Gomory cuts revisited. *Operations Research Letters*, v. 19, n. 1, p. 1 – 9, 1996.

BALAS, E.; FISCHETTI, M.; PULLEYBLANK, W. R. The precedence-constrained asymetric travelling salesman polytope. *Mathematical Programming*, v. 68, p. 214–265, 1995.

BALDACCI, R.; BARTOLINI, E.; MINGOZZI, A.; ROBERTI, R. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, v. 7, n. 3, p. 229–268, 2010.

BALDACCI, R.; BERTOLINI, E.; MINGOZZI, A. An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, v. 59, n. 2, p. 414–426, 2011.

BALINSKI, M. L.; QUANDT, L. On an integer program for a delivery problem. *Operations Research*, v. 12, p. 300–304, 1964.

BALLOU, R. H. (Ed.). *Logística empresarial: transportes, administração de materiais e distribuição física*. São Paulo, Brasil: Atlas, 2010.

BARNHART, C.; LAPORTE, G. *Transportation (In Handbook in operations research and management science.)*. North-Holland, Amsterdam: Elsevier, 2007.

BELENGUER, J.-M.; BENAVENT, E.; PRINS, C.; PRODHON, C.; CALVO, R. W. A branch-and-cut method for the capacitated location-routing problem. *Computers & Operations Research*, v. 38, n. 6, p. 931 – 941, 2011.

BELFIORE, P.; YOSHIZAKI, H. T. Y. Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in brazil. *European Journal of Operational Research*, v. 199, n. 3, p. 750 – 758, 2009.

BERBEGLIA, G.; CORDEAU, J. F.; GRIBKOVSKAIA, I.; LAPORTE, G. Static pickup and delivery problems: a classification scheme and survey. *TOP*, v. 15, n. 1, p. 1–31, 2007.

BERBEGLIA, G.; PESANT, G.; ROUSSEAU, L.-M. Feasibility of the pickup and delivery problem with fixed partial routes: A complexity analysis. *Transportation Science*, v. 46, n. 3, p. 359–373, 2012.

BERTRAND, J. W. M.; FRANSOO, J. C. Operations management research methodologies using quantitative modeling. *International Journal of Operations and Production*, v. 2, n. 2, p. 241–264, 2002.

BETTINELLI, A.; CESELLI, A.; RIGHINI, G. A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, v. 19, n. 5, p. 723 – 740, 2011.

BETTINELLI, A.; CESELLI, A.; RIGHINI, G. A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows. *Mathematical Programming Computation*, v. 6, n. 2, p. 171–197, 2014.

BIANCHESSI, N.; RIGHINI, G. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, v. 34, n. 2, p. 578 – 594, 2007.

BRANCHINI, R. M.; ARMENTANO, V. A.; LØKKETANGEN, A. Adaptive granular local search heuristic for a dynamic vehicle routing problem. *Computers & Operations Research*, v. 36, n. 11, p. 2955 – 2968, 2009.

BRÄYSY, O.; DULLAERT, W.; GENDREAU, M. Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics*, v. 10, p. 587–611, 2005b.

BRÄYSY, O.; GENDREAU, M. Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation Science*, v. 39, n. 1, p. 104–118, 2005.

BRØNMO, G.; NYGREEN, B.; LYSGAARD, J. Column generation approaches to ship scheduling with flexible cargo sizes. *European Journal of Operational Research*, v. 200, n. 1, p. 139 – 150, 2010.

BROWN, G.; GRAVES, G.; RONEN, D. Scheduling ocean transportation of crude oil. *Management Science*, v. 33, p. 335–346, 1987.

CAPRARA, A.; FISCHETTI, M. *Branch-and-cut algorithms (In Annotated Bibliographies in Combinatorial Optimization. Dell' Amico, M. and Maffioli, F. and Martello, S.).* Chichester: Wiley, 1997. 45-64 p.

CHERKESLY, M.; DESAULNIERS, G.; IRNICH, S.; LAPORTE, G. Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and multiple stacks. *European Journal of Operational Research*, p. –, 2015.

CHERKESLY, M.; DESAULNIERS, G.; LAPORTE, G. Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science*, v. 49, n. 4, p. 752–766, 2015.

CHRISTIANSEN, M. Decomposition of a combined inventory and time constrained ship routing problem. *Transportation Science*, v. 33, n. 1, p. 1–14, 1999.

CHRISTIANSEN, M.; FAGERHOLT, K.; RONEN, D. Ship routing and scheduling: status and perspectives. *Transportation Science*, v. 38, n. 1, p. 1–18, 2004.

CHRISTIANSEN, M.; FAGERHOLT, K.; RONEN, D.; NYGREEN, B. *Maritime transportation (In Handbook in operations research and management science. Cynthia Barnhart and Gilbert Laporte.).* Amsterdam, Holland: Elsevier, 2007. 189-284 p.

CLARKE, G.; WRIGHT, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, v. 12, n. 4, p. 568–581, 1964.

CORDEAU, J. F. Branch and cut algorithm for the dial-a-ride problem. *Operation Research*, v. 54, n. 3, p. 573–586, 2006.

CORDEAU, J.-F.; LAPORTE, G. The dial-a-ride problem (DARP): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, v. 1, n. 2, p. 89–101, 2003.

CORDEAU, J.-F.; LAPORTE, G.; ROPKE, S. Recent models and algorithms for one-to-one pickup and delivery problems. In: GOLDEN, B.; RAGHAVAN, S.; WASIL, E. (Ed.). *The Vehicle Routing Problem: Latest Advances and New Challenges*. New York: Springer, 2008, (Operations Research/Computer Science Interfaces, v. 43). p. 327–357.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to Algorithms*. Cambridge, Mass.: The MIT Press, 1990.

CULTURAMIX. *Plataformas offshore*. 2013. Disponível em: <http://www.culturamix.com>.

DABIA, S.; ROPKE, S.; WOENSEL, T. V.; KOK, T. Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, 2012.

DANTZIG, G.; FULKERSON, D. Minimizing the number of tankers to meet a fixed schedule. *Naval Research Logistics Quarterly*, v. 1, p. 217–222, 1954.

DANTZIG, G. B.; WOLFE, P. Decomposition principle for linear programs. *Operations Research*, v. 8, n. 1, p. 101–111, 1960.

DESAULNIERS, G.; DESROSIERS, J.; ERDMANN, A.; SOLOMON, M. M. *VRP with pickup and delivery (In The vehicle routing problem. In Toth, P and Vigo, D.)*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002. 225-242 p.

DESROCHERS, M.; LAPORTE, G. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operational Research Letters*, v. 10, p. 27–36, 1991.

DESROCHERS, M.; LENSTRA, J. K.; SAVELSBERGH, M. W. P.; SOUMIS, F. *Vehicle Routing with Time Windows: Optimization and Approximation (In Vehicle routing: methods and studies. Golden, B. L. and Assad, A. A.)*. North Holland: Elsevier, 1988. 65-84 p.

DESROCHERS, M.; LENSTRA, J. K.; SAVESBERGH, M. W. P. A classifications scheme for vehicle routing and scheduling problems. *Journal of Operational Research Society*, v. 46, p. 322–332, 1990.

DESROSIERS, J.; LÜBBECKE, M. E. *Branch-and-Price-and-cut algorithms. (In Wiley Encyclopedia of Operations Research and Management Science. Cochran, James J. and Cox, Louis A. and Keskinocak, Pinar and Kharoufeh, Jeffrey P. and Smith, J. Cole)*. Chichester: John Wiley & Sons, 2010.

DUMAS, Y.; DESROSIERS, J.; SOUMIS, F. The pickup and delivery problem with time windows. *European Journal of Operational Research*, v. 54, n. 1, p. 7 – 22, 1991.

DUMITRESCU, I.; ROPKE, S.; CORDEAU, J.-F.; LAPORTE, G. The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm. *Mathematical Programming*, v. 121, n. 2, p. 269–305, 2010.

EKSIOGLU, B.; VURAL, A. V.; REISMAN, A. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, v. 57, n. 4, p. 1472 – 1483, 2009.

FAGERHOLT, K. Evaluating the trad-off between the level of customer service and transportation cost in a ship scheduling problem. *Maritime Policy Management*, v. 27(2), p. 145–153, 2000.

FAGERHOLT, K. Ship scheduling with soft time windows: an optimisation based approach. *European Journal of Operational Research*, v. 20, 2001.

FAGERHOLT, K.; RONEN, D. Bulk ship routing and scheduling: solving practical problems may provide better results. *Maritime Policy & Management: The flagship journal of international shipping and port research*, v. 40, n. 1, p. 48–64, 2013.

FEILLET, D.; DEJAX, P.; GENDREAU, M.; GUEGUEN, C. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, Wiley Subscription Services, Inc., A Wiley Company, v. 44, n. 3, p. 216–229, 2004.

FISHER, M. L.; JAIKUMAR, R. A generalized assignment heuristic for vehicle routing. *Networks*, v. 11, n. 2, p. 109–124, 1981.

FLEISCHMANN, B. A new class of cutting planes for the symmetric travelling salesman problem. *Mathematical Programming*, v. 40, n. 1-3, p. 225–246, 1988.

FURTADO, M. G.; MUNARI, P.; MORABITO, R. Pickup and delivery problem with time windows: a new compact two-index formulation. *Submetido para publicação*, 2015.

GAVISH, B.; GRAVES, S. C. The travelling salesman problem and related problems. *Working Paper 7905*, Graduate School of Management, University of Rochester, 1979.

GAVISH, B.; GRAVES, S. C. Scheduling and routing in transportation and distributions systems: Formulations and new relaxations. *Working Paper*, Graduate School of Management, University of Rochester, 1982.

GENDREAU, M.; LAPORTE, G.; VIGO, D. Heuristics for the traveling salesman problem with pickup and delivery. *Computers & Operations Research*, v. 26, n. 7, p. 699 – 714, 1999.

GOLDEN, B.; RAGHAVAN, S.; WASIL, E. *The Vehicle Routing Problem: Latest Advances and New Challenges*. New York: Springer, 2008.

GONDZIO, J.; GONZÁLEZ-BREVIS, P.; MUNARI, P. New developments in the primaldual column generation technique. *European Journal of Operational Research*, v. 224, n. 1, p. 41–51, 2013.

GRIBKOVSKAIA, I.; LAPORTE, G.; SHLOPAK, A. A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms. *Journal of the Operational Research Society*, v. 59, p. 1449–1459, 2008.

GRØNHAUG, R.; CHRISTIANSEN, M.; DESAULNIERS, G.; DESROSIERS, J. A branch-and-price method for a liquefied natural gas inventory routing problem. *Transportation Science*, v. 44, n. 3, p. 400–415, 2010.

GRÖTSCHEL, M.; PADBERG, M. W. On the symmetric travelling salesman problem I: Inequalities. *Mathematical Programming*, v. 16, n. 1, p. 265–280, 1979.

GRÖTSCHEL, M.; PADBERG, M. W. *Polyhedral theory (In The travelling salesman problem In Lawler, E. L. and Lenstra, J. K. and Rinnooy Kan, A. H. G. and Shmoys, D. B.)*. Chichester: Springer, 1985. 251-305 p.

GUTIÉRREZ-JARPA, G.; DESAULNIERS, G.; LAPORTE, G.; MARIANOV, V. A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research*, v. 206, n. 2, p. 341 – 349, 2010.

HENNIG, F.; NYGREEN, B.; CHRISTIANSEN, M.; FAGERHOLT, K.; FURMAN, K. C.; SONG, J.; KOCIS, G. R.; WARRICK, P. Maritime crude oil transportation - a split pickup and split delivery problem. *European Journal of Operational Research*, v. 218, p. 764–774, 2012.

HOFF, A.; ANDERSSON, H.; CHRISTIANSEN, M.; HASLE, G.; LØKKETANGEN, A. Industrial aspects and literature survey: fleet composition and routing. *Computers and Operations Research*, v. 37, p. 2041–2061, 2010.

HWANG, H.; VISOLDILOKPUN, S.; ROSENBERGER, J. M. A branch-and-price-and-cut method for ship scheduling with limited risk. *Transportation Science*, v. 42, n. 3, p. 336–351, 2008.

ILOG. *Informações técnicas*. 2013. Disponível em: <http://www.ilog.com/products/cplex>.

KARAOGLAN, I.; ALTIPARMAK, F.; KARA, I.; DENGIZ, B. A branch and cut algorithm for the location-routing problem with simultaneous pickup and delivery. *European Journal of Operational Research*, v. 211, n. 2, p. 318 – 332, 2011.

KOBAYASHI, K.; KUBO, M. Optimization of oil tanker schedules by decomposition column generation, and time-space network techniques. *Japan Journal of Industrial Applied Mathematics*, v. 27, p. 161–173, 2010.

KOÇ, Ç.; BEKTAS, T.; JABALI, O.; LAPORTE, G. Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research*, p. –, 2015.

KÜÇÜKOGLU, I.; ÖZTÜRK, N. A differential evolution approach for the vehicle routing problem with backhauls and time windows. *Journal of Advanced Transportation*, 2013.

LAPORTE, G. Fifty years of vehicle routing. *Transportation Science*, v. 43, n. 4, p. 408–416, 2009.

LAPORTE, G.; MERCURE, H.; NOBERT, Y. An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, v. 16, n. 1, p. 33–46, 1986.

LAPORTE, G.; NOBERT, Y. *Exact algorithms for the vehicle routing problem (In Surveys in Combinatorial Optimization. Martello, S. and Laporte, G. and Minoux, M.)*. North-Holland, Amsterdam: Ribeiro, 1987. 147-184 p.

LETCHFORD, A. N.; LYSGAARD, J.; EGLESE, R. W. A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Society*, v. 58, p. 1642–1651, 2007.

LU, Q.; DESSOUKY, M. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, v. 38, n. 4, p. 503–514, 2004.

LÜBBECKE, M. E.; DESROSIERS, J. Selected topics in column generation. *Operations Research*, v. 53, n. 6, p. 1007–1023, 2005.

LYSGAARD, J. Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research*, v. 175, n. 1, p. 210 – 223, 2006.

LYSGAARD, J.; LETCHFORD, A. N.; EGLESE, R. W. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, v. 100, p. 423–445, 2004.

MARTELLO, S.; TOTH, P. *Knapsack problems: algorithms and computer implementations*. New York, NY, USA: [s.n.], 1990. ISBN 0-471-92420-2.

MCKAY, M.; HARTLEY, H. Computerized scheduling of seagoing tankers. *Naval Research Logistics*, v. 21, p. 255–264, 1974.

MDIC. *Ministério do Desenvolvimento, Indústria e Comério Exterior*. 2011. Disponível em: <http://www.mdic.gov.br/sitio/>.

MOCCIA, L.; CORDEAU, J.-F.; GAUDIOSO, M.; LAPORTE, G. A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. *Naval Research Logistics*, v. 53, n. 1, p. 45–59, 2006.

MORABITO, R.; PUREZA, V. *Modelagem e Simulação (In Metodologia de pesquisa em engenharia de produção e gestão de operações. Miguel, A. M.)*. Rio de Janeiro, Brasil: Elsevier, 2010. 165-194 p.

MORENO, L.; ARAGÃO, M. P. de; UCHOA, E. Improved lower bounds for the split delivery vehicle routing problem. *Operations Research Letters*, v. 38, n. 4, p. 302 – 306, 2010.

MUNARI, P.; GONDZIO, J. Using the primal-dual interior point algorithm within the branch-and-price-and-cut method. *Computers and Operations Research*, v. 40, p. 2026–2036, 2013.

NADDEF, D.; RINALDI, G. *Branch-and-cut algorithms for the capacitated VRP (In The vehicle routing problem. Toth, P. and Vigo, D.)*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002. 225-242 p.

NAGY, G.; SALHI, S. Location-routing: Issues, models and methods. *European Journal of Operational Research*, v. 177, n. 2, p. 649 – 672, 2007.

NOWAK, M.; ERGUN, O.; White III, C. C. Pickup and delivery with split loads. *Transportation Science*, v. 42, n. 1, p. 32–43, 2008.

NOWAK, M.; ERGUN, O.; White III, C. C. An empirical study on the benefit of split loads with the pickup and delivery problem. *European Journal of Operational Research*, v. 198, n. 3, p. 734 – 740, 2009.

PADBERG, M.; RINALDI, G. A branch-and-cut algorithm for the resolution of large-scale symmetric travelling salesman problemas. *Society for Industrial and Applied Mathematics*, v. 33, n. 1, p. 60–100, 1991.

PARRAGH, S.; DOERNER, K.; HARTL, R. A survey on pickup and delivery problems. part I: transportation between customers and depot. *Journal fur Betriebswirtschaft*, v. 58, n. 1, p. 21–51, 2008.

PARRAGH, S.; DOERNER, K.; HARTL, R. A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. *Journal fur Betriebswirtschaft*, v. 58, n. 2, p. 81–117, 2008b.

PENNA, P. H. V.; SUBRAMANIAN, A.; OCHI, L. S. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, v. 19, n. 2, p. 201–232, 2013.

PEREIRA, F.; TAVARES, J. *Bio-inspired Algorithms for the Vehicle Routing Problem*. Berlin: Springer, 2008.

PERSSON, J.; GÖTHE-LUNDGREN, M. Shipment planning at oil refineries using column generation and valid inequalities. *European Journal of Operational Research*, v. 163, p. 631–652, 2005.

PESSOA, A.; ARAGÃO, M. P.; UCHOA, E. *Robust branch-and-cut-and-price algorithms for vehicle routing problems (In The vehicle routing problem: latest advances and new challenges. In Golden, B. and Raghavan, S. and Wasil, E.)*. New York, USA: Springer, 2008. 297-325 p.

PESSOA, A.; UCHOA, E.; ARAGÃO, M. Poggi de. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Networks*, v. 54, n. 4, p. 167–177, 2009.

PETROBRAS. *Relatório anual*. 2012. Disponível em: <http://www.petrobras.com.br>.

PETROBRAS. *Quem somos - perfil de atividades*. 2013. Disponível em: <http://www.petrobras.com.br/pt/quem-somos/perfil/atividades/>.

PILLAC, V.; GENDREAU, M.; GUÉRET, C.; MEDAGLIA, A. L. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, v. 225, n. 1, p. 1 – 11, 2013.

POTVIN, J. Y. State-of-the-art review evolutionary algorithms for vehicle routing. *Journal on Computing*, v. 21, n. 4, p. 518–548, 2009.

PUREZA, V.; FRANÇA, P. M. Uma abordagem adaptativa de busca tabu aplicada ao problema de roteamento de veículos. *Transportes*, Rio de Janeiro, v. 9, n. 2, p. 28–47, 2001.

PUREZA, V.; LAPORTE, G. Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. *INFOR: Information Systems and Operational Research*, v. 46, p. 165–176, 2008.

PUREZA, V.; MORABITO, R.; REIMANN, M. Vehicle routing with multiple deliverymen: Modeling and heuristic approaches for the {VRPTW}. *European Journal of Operational Research*, v. 218, n. 3, p. 636 – 647, 2012.

QU, Y.; BARD, J. F. A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. *Transportation Science*, v. 49, n. 2, p. 254–270, 2015.

REINHARDT, L.; PISINGER, D. A branch and cut algorithm for the container shipping network design problem. *Flexible Services and Manufacturing Journal*, v. 24, n. 3, p. 349–374, 2012.

REIS, J.; CUNHA, C. Uma nova heurística baseada na meta-heurística VNS para o problema de agrupamento de entregas em veículos de uma frota heterogênea. *Transportes*, v. 19, n. 3, 2011.

RIGHINI, G.; SALANI, M. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, v. 3, n. 3, p. 255 – 273, 2006.

ROCHA, R.; GROSSMANN, I. E.; ARAGÃO, M. V. S. P. Petroleum allocation at PETROBRAS: Mathematical model and a solution algorithm. *Computers and Chemical Engineering*, v. 33, p. 2123–2133, 2009.

RODRIGUES, V. P. *Uma abordagem de otimização para a roteirização e programação de navios: um estudo de caso na indústria petrolífera*. 2013. Monografia de Qualificação de Mestrado, Departamento de Engenharia de Produção - Universidade Federal de São Carlos.

RONEN, D. Cargo ships routing and scheduling: Survey of models and problems. *European Journal of Operational Research*, v. 12, p. 119–126, 1983.

RONEN, D. Ship scheduling: the last decade. *European Journal of Operational Research*, v. 71, p. 325–333, 1993.

ROPKE, S.; CORDEAU, J. F. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, v. 43, n. 3, p. 267–286, 2009.

ROPKE, S.; CORDEAU, J. F.; LAPORTE, G. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, v. 49, n. 4, p. 258–272, 2007.

ROPKE, S.; PISINGER, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, v. 40, n. 4, p. 455–472, 2006.

RULAND, K. S.; RODIN, E. Y. The pickup and delivery problem: faces and branch-and-cut algorithm. *Computers & Mathematics with Applications*, v. 33, p. 1–13, 1997.

SALANI, M.; VACCA, I. Branch and price for the vehicle routing problem with discrete split deliveries and time windows. *European Journal of Operational Research*, v. 213, n. 3, p. 470 – 477, 2011.

SAVELSBERGH, M. W. P.; SOL, M. The general pickup and delivery problem. *Transportation Science*, v. 29, n. 1, p. 17–29, 1995.

SEIXAS, M. P. *Heuristic and exact methods applied to a rich vehicle routing and scheduling problem*. Tese (Doutorado) — POLI-USP, 2013.

SHERALI, H.; AL-YAKOOB, S.; HASSAN, M. Fleet management models and algorithms for an oil-tanker routing and scheduling problem. *IIE transactions*, v. 31, p. 395–406, 1999.

SIGURD, M.; PISINGER, D.; SIG, M. Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science*, v. 38, n. 2, p. 197–209, 2004.

SOL, M. *Column generation techniques for pickup and delivery problems*. Tese (Doutorado) — Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 1994.

STÅLHANE, M.; ANDERSSON, H.; CHRISTIANSEN, M.; CORDEAU, J.-F.; DESAULNIERS, G. A branch-price-and-cut method for a ship routing and scheduling problem with split loads. *Computers & Operations Research*, v. 39, n. 12, p. 3361 – 3375, 2012.

SUBRAMANIAN, A.; UCHOA, E.; OCHI, L. S. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, v. 40, n. 10, p. 2519 – 2531, 2013.

TAS, D.; GENDREAU, M.; DELLAERT, N.; van Woensel, T.; KOK, A. de. Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. *European Journal of Operational Research*, 2013. Disponível em: <http://dx.doi.org/10.1016/j.ejor.2013.05.024>.

TOTH, P.; VIGO, D. (Ed.). *The vehicle routing problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002. ISBN 0-89871-498-2.

TOTH, P.; VIGO, D. *Branch-and-Bound Algorithms for the Capacitated VRP (In The vehicle routing problem. In Toth, P and Vigo, D.)*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002c. 29-51 p.

TOTH, P.; VIGO, D. (Ed.). *Vehicle Routing: Problems, Methods and Applications, 2nd edition*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2014.

VANDERBECK, F. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, v. 48, n. 1, p. 111–128, 2000.

VATN, K. D. *Optimization of water-borne crude oil transport*. Dissertação (Mestrado) — Department of Engineering Cybernetics - Norwegian University of Science and Technology, 2007.

WOLSEY, L. A. *Integer Programming*. New York: John Wiley, 1998.

XU, H.; CHEN, Z.-L.; RAJAGOPAL, S.; ARUNAPURAM, S. Solving a practical pickup and delivery problem. *Transportation Science*, v. 37, n. 3, p. 347–364, 2003.

YAMAN, H. Formulations and valid inequalities for the heterogeneous vehicle routing problem. *Mathematical Programming*, v. 106, n. 2, p. 365–390, 2006.

APÊNDICE A – Restrições de Alcance

Segue abaixo todas as possibilidades que devem ser analisadas para que um arco entre ou não no conjunto A^-.

•Se i,j,k são nós de coleta (PPP):

–Os três nós precisam ser diferentes;

$*(s_0,...,i,j,...,k)$ e capacidade $q_i + q_j + q_k \leq Cap$;

$*(s_0,...,i,j,...,i+n,...,j+n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_k \leq Cap$;

$*(s_0,...,i,j,...,j+n,...,i+n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_k \leq Cap$;

$*(s_0,...,i,j,...,i+n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_j + q_k \leq Cap$;

$*(s_0,...,i,j,...,j+n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_i + q_k \leq Cap$.

•O nó i é de entrega e os nós j e k são de coleta (DPP):

–$i - n \neq j$ e $i - n \neq k$;

$*(s_0,...,i-n,...,i,j,...k)$ e capacidade: $q_{i-n} \leq Cap$ e $q_j + q_k \leq Cap$;

$*(s_0,...,i-n,...,i,j,...,j+n,...k)$ e capacidade: $q_{i-n} \leq Cap$ e $q_j \leq Cap$ e $q_k \leq Cap$.

•Os nós i e j são nós de entrega e o nó k é nó de coleta (DDP):

–$i - n \neq k$ e $j - n \neq k$;

Duas opções de janela de tempo:

$*(s_0,...,i-n,...,j-n,...i,j,...k)$ e capacidade $q_{i-n} + q_{j-n} \leq Cap$ e $q_k \leq Cap$;

$*(s_0,...,j-n,...,i-n,...i,j,...k)$ e capacidade $q_{i-n} + q_{j-n} \leq Cap$ e $q_k \leq Cap$.

•Os três nós são de entrega (DDD):

–Os três nós precisam ser diferentes;

Temos que analisar 8 opções de janela de tempo:

$*(s_0,...,i-n,...,j-n,...k-n,...i,j,...k)$ e capacidade $q_{i-n}+q_{j-n}+q_{k-n} \leq Cap$;

$*(s_0,...,i-n,...,k-n,...j-n,...i,j,...k)$ e capacidade $q_{i-n}+q_{j-n}+q_{k-n} \leq Cap$;

$*(s_0,...,k-n,...,i-n,...j-n,...i,j,...k)$ e capacidade $q_{i-n}+q_{j-n}+q_{k-n} \leq Cap$;

$*(s_0,...,k-n,...,j-n,...i-n,...i,j,...k)$ e capacidade $q_{i-n}+q_{j-n}+q_{k-n} \leq Cap$;

$*(s_0,...,j-n,...,k-n,...i-n,...i,j,...k)$ e capacidade $q_{i-n}+q_{j-n}+q_{k-n} \leq Cap$;

$*(s_0,...,j-n,...,i-n,...k-n,...i,j,...k)$ e capacidade $q_{i-n}+q_{j-n}+q_{k-n} \leq Cap$;

$*(s_0,...,j-n,...,i-n,...i,j,,...,k-n,...k)$ e capacidade $q_{i-n}+q_{j-n} \leq Cap$ e $q_{k-n} \leq Cap$;

$*(s_0,...,i-n,...,j-n,...i,j,,...,k-n,...k)$ e capacidade $q_{i-n}+q_{j-n} \leq Cap$ e $q_{k-n} \leq Cap$.

•O nó i é de coleta e os nós j e k são de entrega (PDD):

–Se $j-n \neq i$ e $k-n \neq j$:

$*(s_0,...,j-n,,...,k-n,...,i,j,...,k)$ e capacidade $q_i+q_{j-n}+q_{k-n} \leq Cap$;

$*(s_0,...,k-n,...,j-n,...,i,j,...,k)$ e capacidade $q_i+q_{j-n}+q_{k-n} \leq Cap$;

$*(s_0,...,j-n,...,i,j,...,k-n,...,k)$ e capacidade $q_i+q_{j-n} \leq Cap$ e $q_{k-n}+q_i \leq Cap$;

$*(s_0,...,j-n,...,k-n,...,i,j,...,i+n,...,k)$ e capacidade $q_i+q_{j-n}+q_{k-n} \leq Cap$;

$*(s_0,...,k-n,...,j-n,...,i,j,...,i+n,...,k)$ e capacidade $q_i+q_{j-n}+q_{k-n} \leq Cap$;

$*(s_0,...,j-n,...,i,j,...,i+n,...,k-n,...,k)$ e capacidade $q_i+q_{j-n} \leq Cap$ e $q_{k-n} \leq Cap$;

$*(s_0,...,j-n,...,i,j,...,k-n,...,i+n,...,k)$ e capacidade $q_i+q_{j-n} \leq Cap$ e $q_{k-n}+q_i \leq Cap$.

–Se $j - n = i$:

Temos duas possibilidades:

$*(s_0,...,k-n,...,i,j,...,k)$ e capacidade $q_i + q_{k-n} \leq Cap$;

$*(s_0,...,i,j,...,k-n,...,k)$ e capacidade $q_i \leq Cap$ e $q_{k-n} \leq Cap$.

–$k - n = i$:

$*(s_0,...,j-n,...,i,j,...,k)$ e capacidade $q_i + q_{j-n} \leq Cap$.

•Os nós i e j são de coleta e o nó k é de entrega (PPD):

–Se $k - n \neq i$ e $k - n \neq j$:

$*(s_0,...,k-n,...,i,j,...,k)$ e capacidade $q_i + q_j + q_{k-n} \leq Cap$;

$*(s_0,...,i,j,...,k-n,...,k)$ e capacidade $q_i + q_j + q_{k-n} \leq Cap$;

$*(s_0,...,k-n,...,i,j,...,i+n,...,j+n,...,k)$ e capacidade $q_i + q_j + q_{k-n} \leq Cap$;

$*(s_0,...,k-n,...,i,j,...,j+n,...,i+n,...,k)$ e capacidade $q_i + q_j + q_{k-n} \leq Cap$;

$*(s_0,...,i,j,...,k-n,...,j+n,...,i+n,...,k)$ e capacidade $q_i + q_j + q_{k-n} \leq Cap$;

$*(s_0,...,i,j,...,k-n,...,i+n,...,j+n,...,k)$ e capacidade $q_i + q_j + q_{k-n} \leq Cap$;

$*(s_0,...,i,j,...,i+n,...,k-n,...,j+n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_j + q_{k-n} \leq Cap$;

$*(s_0,...,i,j,...,i+n,...,j+n,...,k-n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_{k-n} \leq Cap$;

$*(s_0,...,i,j,...,j+n,...,i+n,...,k-n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_{k-n} \leq Cap$;

$*(s_0,...,i,j,...,j+n,...,k-n,...,i+n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_{k-n} + q_i \leq Cap$;

$*(s_0,...,k-n,...,i,j,...,i+n,...,k)$ e capacidade $q_i + q_j + q_{k-n} \leq Cap$;

$*(s_0,...,k-n,...,i,j,...,j+n,...,k)$ e capacidade $q_i + q_j + q_{k-n} \leq Cap$;

$*(s_0,...,i,j,...,i+n,...,k-n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_{k-n} + q_j \leq Cap$;

$*(s_0,...,i,j,...,k-n,...,i+n,...,k)$ e capacidade $q_i + q_j + q_{k-n} \leq Cap$;

$*(s_0,...,i,j,...,j+n,...,k-n,...,k)$ e capacidade $q_i + q_j \leq Cap$ e $q_{k-n} + q_i \leq Cap$;

$*(s_0,...,i,j,...,k-n,...,j+n,...,k)$ e capacidade $q_i + q_j + q_{k-n} \leq Cap$.

–Se $k - n = i$:

$*(s_0,...,i,j,...,j+n,...,k)$ e capacidade $q_i + q_j \leq Cap$;

$*(s_0,...,i,j,...,k)$ e capacidade $q_i + q_j \leq Cap$.

–Se $k - n = j$:

$*(s_0,...,i,j,...,i+n,...,k)$ e capacidade $q_i + q_j \leq Cap$;

$*(s_0,...,i,j,...,k)$ e capacidade $q_i + q_j \leq Cap$.

•Os nós i e k são nós de coleta e o nó j é um nó de entrega (PDP):

–Se $j - n \neq k$ e $j - n \neq i$:

$*(s_0,...,j-n,...,i,j,...,k)$ e capacidade $q_i + q_k \leq Cap$ e $q_i + q_{j-n} \leq Cap$;

$*(s_0,...,j-n,...,i,j,...,i+n,...,k)$ e capacidade $q_i + q_{j-n} \leq Cap$ e $q_k \leq Cap$.

–Se $j - n = i$ e $j - n \neq k$:

Temos apenas uma opção para analisar:

$*(s_0,...,i,j,...,k)$ e capacidade $q_i \leq Cap$ e $q_k \leq Cap$.

•Os nós i e k são nós de entrega e o nó j é nó de coleta (DPD):

–Se $i - n \neq j$ e $k - n \neq j$:

$*(s_0,...,i-n,...,k-n,...,i,j,...,k)$ e capacidade $q_{i-n} + q_{k-n} \leq Cap$ e $q_j + q_{k-n} \leq Cap$;

$*(s_0,...,k-n,...,i-n,...,i,j,...,k)$ e capacidade $q_{i-n} + q_{k-n} \leq Cap$ e $q_j + q_{k-n} \leq Cap$;

$*(s_0,...,i-n,...,i,j,....,k-n,...,k)$ e capacidade $q_{i-n} \leq Cap$ e $q_j + q_{k-n} \leq Cap$;

$*(s_0,...,i-n,...,k-n,...,i,j,....,j+n,...,k)$ e capacidade $q_{i-n} + q_{k-n} \leq Cap$ e $q_j + q_{k-n} \leq Cap$;

$*(s_0,...,k-n,...,i-n,...,i,j,....,j+n,...,k)$ e capacidade $q_{i-n} + q_{k-n} \leq Cap$ e $q_j + q_{k-n} \leq Cap$;

$*(s_0,...,i-n,...,i,j,....,j+n,...,k-n,...,k)$ e capacidade $q_{i-n} \leq Cap$ e $q_j \leq Cap$ e $q_{k-n} \leq Cap$;

$*(s_0,...,i-n,...,i,j,....,k-n,...,j+n,...,k)$ e capacidade $q_{i-n} \leq Cap$ e $q_j + q_{k-n} \leq Cap$.

–Se $i - n \neq j$ e $k - n = j$:

Apenas uma opção para analisar:

$*(s_0,...,i-n,...,i,j,...,k)$ e capacidade $q_{i-n} \leq Cap$ e $q_j \leq Cap$.

Temos que analisar agora quando o nó $i = s_0$, ou seja, se for o depósito inicial:

•$i = s_0$, j é um nó de coleta e k é um nó de entrega (s_0PD):

–Se $k - n \neq j$:

$*(s_0,j,...,k-n,...,k)$ e capacidade $q_j + q_{k-n} \leq Cap$;

$*(s_0,j,...,j+n,...,k-n,...,k)$ e capacidade $q_j \leq Cap$ e $q_{k-n} \leq Cap$;

$*(s_0,j,...,k-n,...,j+n,...,k)$ e capacidade $q_j + q_{k-n} \leq Cap$.

–Se $k - n = j$:

Apenas uma opção:

$*$Janela de tempo: $(s_0,j,....,k)$ e capacidade $q_j \leq Cap$.

•$i = s_0$, j e k são nós de coleta (s_0PP):

–$j \neq k$;

$*(s_0,j,...,k)$ e capacidade $q_j + q_k \leq Cap$;

$*(s_0,j,...,j+n,...,k)$ e capacidade $q_j \leq Cap$ e $q_k \leq Cap$.

Agora, temos que analisar se $j = k$:

•Se o nó i for de coleta e o nó j for de entrega (PD):

–Se $k - n \neq i$:

Apenas uma opção:

$*(s_0,...,k-n,...,i,k)$ e capacidade $q_{k-n} + q_i \leq Cap$.

–Se $k - n = i$:

Apenas uma opção:

$*(s_0,...,i,j)$ e capacidade $q_i \leq Cap$.

•Se os nós i e j são nós de entrega (DD):

–$i \neq j$;

Temos duas opções:

$*(s_0,...,i-n,...j-n,...,i,j)$ e capacidade $q_{i-n}+q_{j-n} \leq Cap$;

$*(s_0,...,j-n,...,i-n,...,i,j)$ e capacidade $q_{i-n}+q_{j-n} \leq Cap$.

•Os nós i e j são nós de coleta (PP):

$-i \neq j$;

Apenas uma opção:

$*(s_0,...,i,j)$ e capacidade $q_i+q_j \leq Cap$.

•O nó i é um nó de entrega e o nó j é um nó de coleta (DP):

$-i-n \neq j$;

Apenas uma opção:

$*(s_0,...,i-n,...,i,j)$ e capacidade $q_{i-n} \leq Cap$ e $q_j \leq Cap$.

•Por último, temos que analisar quando $i = s_0$ e $j = k$ e j é um nó de coleta:

Temos apenas uma opção:

$-(s_0,j)$ e capacidade $q_j \leq Cap$.

Segue abaixo todas as possibilidades que devem ser analisadas para que um arco entre ou não no conjunto A^+.

•Os três nós são nós de coleta (PPP):

$-$Os três nós são diferentes ($i \neq j \neq k$):

$*(k,...,k+n,...i,j,...,j+n,...,i+n,...,en_0)$ e capacidade $q_k \leq Cap$ e $q_i+q_j \leq Cap$;

$*(k,...,k+n,...i,j,...,i+n,...,j+n,...,en_0)$ e capacidade $q_k \leq Cap$ e $q_i+q_j \leq Cap$;

$*(k,...i,j,...,k+n,...,i+n,...,j+n,...,en_0)$ e capacidade $q_i+q_j+q_k \leq Cap$;

$*(k,...i,j,...,k+n,...,j+n,...,i+n,...,en_0)$ e capacidade $q_i+q_j+q_k \leq Cap$;

$*(k,...i,j,...,j+n,...,k+n,...,i+n,...,en_0)$ e capacidade $q_i+q_j+q_k \leq Cap$;

$*(k,...i,j,...,j+n,...,i+n,...,k+n,...,en_0)$ e capacidade $q_i+q_j+q_k \leq Cap$;

$*(k,...i,j,...,i+n,...,j+n,...,k+n,...,en_0)$ e capacidade $q_i+q_j+q_k \leq Cap$;

$*(k,...i,j,...,i+n,...,k+n,...,j+n,...,en_0)$ e capacidade $q_i+q_j+q_k \leq Cap$.

•k é um nó de entrega e i e j são nós de coleta (DPP):

–Os três nós são diferentes $k-n \neq i \neq j$:

∗$(k,...i,j,...,i+n,...,j+n,...,en_0)$ e capacidade $q_{k-n} \leq Cap$ e $q_i + q_j \leq Cap$;

∗$(k,...i,j,...,j+n,...,i+n,...,en_0)$ e capacidade $q_{k-n} \leq Cap$ e $q_i + q_j \leq Cap$.

•Os nós k e i são de entrega e o nó j é de coleta (DDP):

–Os três nós são diferentes: $k-n \neq i-n \neq j$:

∗$(k,...,i-n,...,i,j,...,j+n,...,en_0)$ e capacidade $q_{k-n} \leq Cap$ e $q_{i-n} \leq Cap$ e $q_j \leq Cap$;

∗$(k,...,i,j,...,j+n,...,en_0)$ e capacidade $q_{k-n} + q_{i-n} \leq Cap$ e $q_j \leq Cap$.

•Os três nós são de entrega (DDD):

–Os três nós são diferentes: $k-n \neq i-n \neq j-n$:

∗$(k,...,i,j,...,en_0)$ e capacidade $q_{k-n} + q_{i-n} + q_{j-n} \leq Cap$;

∗$(k,...,i-n,...,j-n,...,i,j,...,en_0)$ e capacidade $q_{k-n} \leq Cap$ e $q_{i-n} + q_{j-n} \leq Cap$;

∗$(k,...,j-n,...,i-n,...,i,j,...,en_0)$ e capacidade $q_{k-n} \leq Cap$ e $q_{i-n} + q_{j-n} \leq Cap$;

∗$(k,...,j-n,...,i,j,...,en_0)$ e capacidade $q_{i-n} + q_{k-n} \leq Cap$ e $q_{i-n} + q_{j-n} \leq Cap$;

∗$(k,...,i-n,...,i,j,...,en_0)$ e capacidade $q_{j-n} + q_{k-n} \leq Cap$ e $q_{j-n} + q_{i-n} \leq Cap$.

•O nó k é um nó de coleta e os nós i e j são nós de entrega (PDD):

–Os três nós são diferentes $k \neq i-n \neq j-n$:

∗$(k,...,i-n,...,j-n,...,i,j,...,k+n,...,en_0)$ e capacidade $q_{i-n} + q_{j-n} + q_k \leq Cap$;

∗$(k,...j-n,...,i-n,...,i,j,...,k+n,...,en_0)$ e capacidade $q_{i-n} + q_{j-n} + q_k \leq Cap$;

∗$(k,...,k+n,...,j-n,...,i-n,...,i,j,...,en_0)$ e capacidade $q_{i-n} + q_{j-n} \leq Cap$ e $q_k \leq Cap$;

$*(k,...,k+n,...,i-n,...,j-n,...,i,j,...,en_0)$ e capacidade $q_{i-n}+q_{j-n} \leq Cap$ e $q_k \leq Cap$;

$*(k,...,i-n,...,j-n,...,k+n,...,i,j,...,en_0)$ e capacidade $q_{i-n}+q_{j-n}+q_k \leq Cap$;

$*(k,...,i-n,...,k+n,...,j-n,...,i,j,...,en_0)$ e capacidade $q_{i-n}+q_k \leq Cap$ e $q_{i-n}+q_{j-n} \leq Cap$;

$*(k,...,j-n,...,k+n,...,i-n,...,i,j,...,en_0)$ e capacidade $q_k+q_{j-n} \leq Cap$ e $q_{j-n}+q_{i-n} \leq Cap$;

$*(k,...,j-n,...,i-n,...,k+n,...,i,j,...,en_0)$ e capacidade $q_{i-n}+q_{j-n}+q_k \leq Cap$;

$*(k,...,i,j,...,k+n,...,en_0)$ e capacidade $q_{i-n}+q_{j-n}+q_k \leq Cap$;

$*(k,...,k+n,...,i,j,...,en_0)$ e capacidade $q_{i-n}+q_{j-n}+q_k \leq Cap$;

$*(k,...k+n,...i-n,...,i,j,...,en_0)$ e capacidade $q_{j-n}+q_k \leq Cap$ e $q_{j-n}+q_{i-n} \leq Cap$;

$*(k,...i-n,...k+n,...,i,j,...,en_0)$ e capacidade $q_{j-n}+q_k+q_{i-n} \leq Cap$;

$*(k,...i-n,...,i,j,...,k+n,...,en_0)$ e capacidade $q_{j-n}+q_k+q_{i-n} \leq Cap$;

$*(k,...k+n,...j-n,...,i,j,...,en_0)$ e capacidade $q_{i-n}+q_k \leq Cap$ e $q_{j-n}+q_{i-n} \leq Cap$;

$*(k,...j-n,...k+n,...,i,j,...,en_0)$ e capacidade $q_{j-n}+q_k+q_{i-n} \leq Cap$;

$*(k,..j-n,...,i,j,...,k+n,...,en_0)$ e capacidade $q_{j-n}+q_k+q_{i-n} \leq Cap$.

–Se $k = i-n$:

$*(k,...j-n,...,i,j,...,en_0)$ e capacidade $q_{j-n}+q_k \leq Cap$;

$*(k,...,i,j,...,en_0)$ e capacidade $q_{j-n}+q_k \leq Cap$.

–Se $k = j-n$:

$*(k,...i-n,...,i,j,...,en_0)$ e capacidade $q_{i-n}+q_k \leq Cap$;

$*(k,...,i,j,...,en_0)$ e capacidade $q_{i-n}+q_k \leq Cap$.

•Os nós k e i são nós de coleta e o nó j é um nó de entrega (PPD):

–Os três nós são diferentes $k \neq i \neq j-n$:

$*(k,...,k+n,...,i,j,...,i+n,...,en_0)$ e capacidade $q_{j-n}+q_k \leq Cap$ e $q_i+q_{j-n} \leq Cap$;

$*(k,...,i,j,...,i+n,...k+n,...,en_0)$ e capacidade $q_{j-n}+q_k+q_i \leq Cap$;

$*(k,...,i,j,...,k+n,...i+n,...,en_0)$ e capacidade $q_{j-n}+q_k+q_i \leq Cap$;

$*(k,...,k+n,...,j-n,...,i,j,...,i+n,...,en_0)$ e capacidade $q_{j-n}+q_i \leq Cap$ e $q_k \leq Cap$;

$*(k,...,j-n,...,k+n,...,i,j,...,i+n,...,en_0)$ e capacidade $q_{j-n}+q_i \leq Cap$ e $q_k+q_{j-n} \leq Cap$;

$*(k,...,j-n,...,i,j,...,i+n,...,k+n,...,en_0)$ e capacidade $q_{j-n}+q_i+q_k \leq Cap$;

$*(k,...,j-n,...,i,j,...,k+n,...,i+n,...,en_0)$ e capacidade $q_{j-n}+q_i+q_k \leq Cap$.

–Se $k = j-n$:

$*(k,...,i,j,...,i+n,...,en_0)$ e capacidade $q_k+q_i \leq Cap$.

–Se $i = j-n$:

$*(k,...,k+n,....,i,j,...,en_0)$ e capacidade $q_k \leq Cap$ e $q_i \leq Cap$;

$*(k,....,i,j,...,k+n,...,en_0)$ e capacidade $q_k+q_i \leq Cap$.

•Os nós k e j são nós de coleta e o nó i é um nó de entrega (PDP):

–Os três nós diferentes: $k \neq j \neq i-n$:

$*(k,...,k+n,...,i,j,...,j+n,...,en_0)$ e capacidade $q_k+q_{i-n} \leq Cap$ e $q_j \leq Cap$;

$*(k,...,i,j,...,j+n,...,k+n,...,en_0)$ e capacidade $q_k+q_{i-n} \leq Cap$ e $q_j+q_k \leq Cap$;

$*(k,...,i,j,...,k+n,...,j+n,...,en_0)$ e capacidade $q_k+q_{i-n} \leq Cap$ e $q_j+q_k \leq Cap$;

$*(k,...,k+n,...,i-n,...,i,j,...,j+n,...,en_0)$ e capacidade $q_k \leq Cap$ e $q_j \leq Cap$ e $q_{i-n} \leq Cap$;

$*(k,...,i-n,...,k+n,...,i,j,...,j+n,...,en_0)$ e capacidade $q_k+q_{i-n} \leq Cap$ e $q_j \leq Cap$;

$*(k,...,i-n,...,i,j,...,j+n,...,k+n,...,en_0)$ e capacidade $q_k+q_{i-n} \leq Cap$ e $q_k+q_j \leq Cap$;

$*(k,...,i-n,...,i,j,...,k+n,...,j+n,...,en_0)$ e capacidade $q_k+q_{i-n} \leq Cap$ e $q_k+q_j \leq Cap$.

–Se $k = i-n$:

$*(k,...,i,j,...,j+n,...,en_0)$ e capacidade $q_k \leq Cap$ e $q_j \leq Cap$.

•Os nós k e j são nós de entrega e o nó i é um nó de coletea (DPD):

–Os três nós são diferentes $k-n \neq i \neq j-n$:

$*(k,...,i,j,...,i+n,...,en_0)$ e capacidade $q_k + q_{j-n} \leq Cap$ e $q_i + q_{j-n} \leq Cap$;

$*(k,...,j-n,...,i,j,...,i+n,...,en_0)$ e capacidade $q_k \leq Cap$ e $q_i + q_{j-n} \leq Cap$,

–Se $i = j-n$:

$*(k,...,i,j,...,en_0)$ e capacidade $q_k \leq Cap$ e $q_i \leq Cap$.

Agora, temos que analisar quando $j = en_0$:

•k é um nó de coleta e i um nó de entrega (PD):

–Se $k \neq i-n$:

$*(k,...,k+n,...,i-n,...,i,j)$ e capacidade $q_k \leq Cap$ e $q_{i-n} \leq Cap$;

$*(k,...,i-n,...,k+n,...,i,j)$ e capacidade $q_k + q_{i-n} \leq Cap$;

$*(k,...,k+n,...,i,j)$ e capacidade $q_k + q_{i-n} \leq Cap$.

–Se $k = i-n$:

$*(k,...,i,j)$ e capacidade $q_k \leq Cap$.

•Se k e i são nós de entrega (DD):

–Os nós dois nós são diferentes $k-n \neq i-n$:

$*(k,...,i,j)$ e capacidade $q_{k-n} + q_{i-n} \leq Cap$;

$*(k,...i-n,...,i,j)$ e capacidade $q_{k-n} \leq Cap$ e $q_{i-n} \leq Cap$.

Agora, precisamos analisar quando $k = i$:

•Se i é um nó de coleta e j é um nó de entrega (PD):

–Se $i \neq j-n$:

$*(i,j,....,i+n,...,en_0)$ e capacidade $q_i + q_{j-n} \leq Cap$.

–Se $i = j-n$:

$*(i,j,...,en_0)$ e capacidade $q_i \leq Cap$.

•Se i e j são nós de entrega (DD):

–Os dois nós são diferentes $i-n \neq j-n$:

$*(i,j,...,en_0)$ e capacidade $q_{i-n} + q_{j-n} \leq Cap$.

•Os dois nós são de coleta (PP):

–Os dois nós são diferentes $i \neq j$:

$*(i,j,....,i+n,....,j+n,...,en_0)$ e capacidade $q_i + q_j \leq Cap$;

$*(i,j,....,j+n,....,i+n,...,en_0)$ e capacidade $q_i + q_j \leq Cap$.

•O nó i é um nó de entrega e o nó j é um nó de coleta (DP):

–Os nós nós são diferentes $i - n \neq j$:

$*(i,j,....,j+n,...,en_0)$ e capacidade $q_{i-n} \leq Cap$ e $q_j \leq Cap$,

•Por último, temos que analisar se $j = en_0$ e $i = k$ e i é um nó de entrega:

–(i,en_0) e capacidade $q_{i-n} \leq Cap$.