FORMULAÇÕES E ALGORITMOS DE SOLUÇÃO PARA O CONJUNTO DOMINANTE 2-CONEXO MÍNIMO

Vinícius Leal do Forte

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Abilio Pereira de Lucena Filho

Rio de Janeiro
Março de 2014

FORMULAÇÕES E ALGORITMOS DE SOLUÇÃO PARA O CONJUNTO DOMINANTE 2-CONEXO MÍNIMO

Vinícius Leal do Forte

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Abilio Pereira de Lucena Filho, Ph.D.

Prof. Nelson Maculan Filho, Ph.D.

Prof. Luidi Gelabert Simonetti, D.Sc.

Prof. Márcia Helena Costa Fampa, Ph.D.

Prof. Yuri Abitbol de Menezes Frota, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2014

Forte, Vinícius Leal do

Formulações e Algoritmos de Solução para o Conjunto Dominante 2-Conexo Mínimo/Vinícius Leal do Forte. – Rio de Janeiro: UFRJ/COPPE, 2014.

XI, 76 p.: il.; 29, 7cm.

Orientador: Abilio Pereira de Lucena Filho

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2014.

Referências Bibliográficas: p. 72 – 75.

1. Conjunto Dominante 2-Conexo Mínimo. 2. Programação Inteira Mista. 3. Otimização Combinatória. I. Lucena Filho, Abilio Pereira de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Aos meus pais.

Agradecimentos

Agradeço aos meus pais pelo amor, carinho e apoio, essenciais para que pudesse atingir os meus objetivos.
A minha noiva Caroline Ferreira, pela atenção, amor, carinho e compreensão que foram fundamentais para a conclusão deste trabalho. Ao professor Abilio Lucena, por sua orientação, pela sua dedicação e por ter compartilhado de seu vasto conhecimento com este aluno.
Ao professor Nelson Maculan, pelos conselhos, pelas ajudas e por compartilhar de sua sabedoria tanto acadêmica como humanitária.
Ao professor Luidi Simonetti, pelas muitas sugestões dadas no exame de qualificação que ajudaram a compor importantes contribuições desta tese.
Aos membros da banca de defesa de tese, pelas sugestões que foram de extrema importância para este trabalho.
Aos professores Flávio Montenegro e José André Brito, por sempre estarem dispostos a ajudar no que fosse necessário.
Aos professores e funcionários do PESC, em especial aos professores Paulo Roberto, Adilson Xavier, Felipe França e Celina Miraglia.
Aos amigos do PESC, Jesus, Pedro, Renan, Viviane, Ana Flávia, Rogério, Michael e Virgínia que com comentários e discussões ajudaram a enriquecer esta tese.
A minha amiga Maria de Fátima, pelas ótimas conversas, pelas ajudas e pelos necessários puxões de orelha.
Aos amigos de longa data, Vítor Setubal e Anderson Oliveira, pelo apoio, estímulo e compreensão.
Por fim, agradeço ao CNPq pela bolsa de doutorado concedida.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

FORMULAÇÕES E ALGORITMOS DE SOLUÇÃO PARA O CONJUNTO DOMINANTE 2-CONEXO MÍNIMO

Vinícius Leal do Forte

Março/2014

Orientador: Abilio Pereira de Lucena Filho

Programa: Engenharia de Sistemas e Computação

Seja $G = (V, E)$ um grafo não-direcionado com um conjunto de vértices V e um conjunto de arestas E. Associado a G, assuma que um conjunto não vazio $W \subset V$ é identificado e que este induz um subgrafo G_W em G. W é denominado um conjunto dominante, se todo vértice de G pertence a W ou tem um vizinho em W. Por sua vez, quando G_W é 2-aresta (resp. 2-vértice) conexo o conjunto dominante W é dito 2-aresta (resp. 2-vértice) conexo. Em vários contextos, teóricos ou práticos, encontrar um conjunto dominante 2-{aresta,vértice} conexo de mínima cardinalidade é do especial interesse, sendo este o tema desta tese. Antes de nossa investigação, nenhuma formulação ou algoritmo de solução exata existia para o problema, na literatura. Aqui, introduzimos sete formulações, para ele e algoritmos de solução exata baseados nessas formulações. Além disso, duas heurísticas primais são também propostas e testadas nesta tese. Da mesma forma, comparações analíticas e computacionais foram conduzidas para as formulações e experimentos computacionais foram efetuados para testar os algoritmos de solução. Como resultado de nossa investigação, instâncias do problema definidas sobre grafos com até 200 vértices foram resolvidos à otimalidade.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

FORMULATIONS AND SOLUTION ALGORITHMS FOR THE MINIMUM 2-CONNECTED DOMINATING SET

Vinícius Leal do Forte

March/2014

Advisor: Abilio Pereira de Lucena Filho

Department: Systems Engineering and Computer Science

Let $G = (V, E)$ be an undirected graph with a set of vertices V and a set of edges E. Assuming that a non empty set $W \subset V$ is given, let G_W be the subgraph it induces in G. W is called a dominating set if every vertex of G belongs to W or has a neighbor vertex in that set. Additionally, if G_W is 2-edge (resp. 2-vertex) connected, the dominating set W is called 2-edge (resp. 2-vertex) connected. In various contexts, practical or theoretical, it is particularly interesting to finding a 2-{edge,vertex} connected dominating set with a cardinality as small as possible, and that is precisely the subject of this thesis. Prior to our investigation, no formulation or exact solution algorithm was found in the literature for the problem. Here, seven different formulations are introduced for it, together with corresponding exact solution algorithms. Furthermore, two primal heuristic are also investigated . Likewise, analytical and computational comparisons are carried out among our formulations while computational experiments were implemented to test the solution algorithms. As a result of our investigation, problem instances defined over graphs with as many as 200 vertices were solved to proven optimality.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nas últimas décadas, a ampla gama de serviços oferecidos através de redes de comunicações ou redes afins e o consequente aumento de tráfego que isso provoca, têm crescido acentuadamente. Dessa maneira, para o enorme número de usuários que se beneficiam desses serviços, o nível de confiabilidade por eles oferecidos é de extrema importância. Em particular, isso se aplica a redes de comunicação sem-fio, redes de sensores sem-fio, redes de comércio móvel e redes de procura e resgate, dentre outras. Como ponto em comum, entre os exemplos citados, nenhuma dessas redes possui uma infra estrutura fixa de comunicação, sendo então chamadas de redes sem-fio *ad-hoc*. Tomando como exemplo as redes de sensores sem-fio, seus sensores podem transferir dados uns com os outros desde que a distância entre os mesmos seja menor ou igual ao alcance de transmissão. Uma estrutura denominada *virtual backbone* (espinha dorsal virtual) foi proposta (ALZOUBI *et al.*, 2002) como uma estrutura básica para esse tipo de rede, e impõe que os dados sejam transferidos apenas entre os nós (ou seja, sensores) dessa estrutura central, ao invés de serem transferidos diretamente entre todos os sensores da rede. Alguns trabalhos (SINHA *et al.*, 2001) mostram que a utilização deste tipo de estrutura pode impactar positivamente na robustez da rede. Neste contexto, um sensor pode apresentar falhas devido a um dano acidental ou a um esgotamento de energia. Da mesma forma, a conexão entre um par de sensores pode ser desfeita devido ao distanciamento entre os mesmos. Assim sendo, é extremamente importante que a rede em questão seja, tanto quanto possível, tolerante à falhas, ou seja, possua uma certa redundância em seu trafego de dados. O problema de desenhar *virtual backbones* tolerantes a falhas pode ser formalizado como um problema de otimização combinatória.

Seja $G = (V, E)$ um grafo simples não-direcionado com um conjunto de vértices V e um conjunto de arestas E. Um subconjunto $W \subseteq V$ é um conjunto dominante de G se todo vértice de V pertence a W ou compartilha uma aresta com um vértice de W. Um conjunto dominante é conexo se o subgrafo G_W que este induz em G é conexo. Por conseguinte, W é 2-aresta (2-vértice) conexo se G_W é 2-aresta (2-vértice)

conexo. Para simplificar a notação, o termo "2-{aresta,vértice}" será utilizado no decorrer deste texto sempre que a escolha do tipo de conexidade for arbitrária. Investigaremos aqui o Problema do Conjunto Dominante 2-{aresta,vértice} Conexo Mínimo (PCD2CM), ou seja, o problema de encontrar um conjunto dominante 2-{aresta,vértice} conexo de G com a menor cardinalidade possível. No contexto de redes de sensores sem-fio, esse problema equivale a encontrar um *virtual backbone* G_W com uma quantidade mínima de sensores e possuindo uma redundância no caso da perda de uma conexão, ou na perda de um sensor, respectivamente para as variantes 2-aresta conexa e 2-vértice conexa, do problema.

Outras aplicações envolvendo conjuntos dominantes conexos podem ser encontradas no ajuste de estratégias de defesa contra ataque de *worms* em redes *peer-to-peer* (LIANG e SENCUN, 2007), no desenho de redes de fibra ótica quando regeneradores de informação são necessários (CHEN *et al.*, 2010), e como modelos para investigar interações proteína-proteína (MILENKOVIĆ *et al.*, 2011). Confiabilidade também é uma questão muito importante no desenho de algumas dessas redes. Quando isto se aplica, impor que um conjunto dominante associado a elas seja 2-conexo é um passo natural a ser dado.

Algumas poucas versões diferentes de nosso problema já foram investigados na literatura. Dentre eles, o Problema do Conjunto Dominante Conexo Mínimo (PCDCM), que tem relação com o Problema da Árvore Geradora Com Número Máximo de Folhas (LUCENA *et al.*, 2010), é a variante que atraiu, até agora, o maior interesse. Contribuições para este problema envolvem formulações de Programação Inteira, como em LUCENA *et al.* (2010), SIMONETTI *et al.* (2011) e CHEN *et al.* (2010), algoritmos aproximativos como em THAI *et al.* (2008), heurísticas como em SIMONETTI *et al.* (2011) e LI *et al.* (2005) e algoritmos de solução exata como em LUCENA *et al.* (2010), SIMONETTI *et al.* (2011) e CHEN *et al.* (2010). Além disso, a variante específica do problema onde é exigido que G_W seja um ciclo de G foi investigado para classes especiais de grafos em COLBOURN *et al.* (1985), PROSKUROWSKI (1981) e SKOWROŃSKA e M. (1991). Para um grafo geral, uma formulação para esse tipo de problema e um algoritmo de solução exata foram recentemente introduzidos em LUCENA *et al.* (2013). Por fim, uma generalização ampla do PCDCM, o Problema do Conjunto M-Dominante e K-Conexo Mínimo, foi investigado em WU *et al.* (2007) e SHANG *et al.* (2008). Este problema requer um conjunto dominante W, K-{aresta,vértice} conexo, tal que qualquer vértice de $V \setminus W$ é dominado por pelo menos M vértices distintos de W. No entanto, até esta data, aparentemente apenas uma heurística (WU *et al.*, 2007) e um algoritmo aproximativo (SHANG *et al.*, 2008) foram propostos para esse problema. Especificamente para o PCD2CM, tanto quanto sabemos, as únicas referências existentes na literatura, até o momento, dizem respeito a um algoritmo aproximativo (WANG

et al., 2009) e uma versão preliminar (vide DO FORTE *et al.* (2013)) dos temas abordados nesta tese, introduzindo formulações e algoritmos de solução exata para o problema.

Neste trabalho propomos sete formulações de Programação Inteira Mista para o PCD2CM, inédita na literatura, desigualdades válidas, duas heurísticas primais e algoritmos exatos para o PCD2CM, em suas variantes 2-aresta e 2-vértice conexa. Duas formulações são baseadas em desigualdades de corte não-direcionados. Outras duas se utilizam de multi-fluxos. Por fim, as três últimas são baseadas em reformulações direcionadas do problema. A força destas formulações, medida em termos dos limitantes duais obtidos com a resolução de suas relaxações lineares, são comparadas neste trabalho, tanto teoricamente quanto computacionalmente. Testes computacionais são também conduzidos para os algoritmos de solução exata resultantes de nossas formulações. Dois conjuntos distintos de instâncias aqui introduzidos foram utilizados nos experimentos, e nossos algoritmos foram capazes de resolver, provando a otimalidade, instâncias do problema com até 200 vértices.

Esta tese está organizada da seguinte forma. No Capítulo 2, inicialmente, são introduzidas as notações que serão utilizadas ao longo do texto e apresentados alguns resultados básicos relativos ao PCD2CM. Em seguida, neste mesmo capítulo, diferentes formulações de Programação Inteira Mista são propostas para ambas as variantes do PCD2CM. Da mesma forma, um conjunto de desigualdades válidas é introduzido neste capítulo, para algumas das formulações propostas. Encerramos o segundo capítulo apresentando demonstrações referentes a relações de dominância entre as formulações propostas. No Capítulo 3, duas heurísticas primais são propostas para ambas as variantes do problema, além de um procedimento de busca local para tentar reduzir a cardinalidade das soluções encontradas. Algoritmos de solução exata são descritos no Capítulo 4. No Capítulo 5 são apresentados os resultados obtidos no experimento computacional conduzido com os mesmos. Comparações dos resultados obtidos pelos diferentes algoritmos são ali efetuadas. Finalmente, no Capítulo 6, são apresentadas algumas conclusões e sugestões para trabalhos futuros.

Capítulo 2

Formulações e Desigualdades Válidas.

Neste capítulo serão apresentadas formulações de Programação Inteira Mista para o PCD2CM. Inicialmente, serão apresentadas formulações para a variante 2-aresta conexa do PCD2CM, bem como serão investigadas possíveis relações de equivalência entre as formulações propostas. Em seguida, formulações para a variante 2-vértice conexa serão apresentadas e suas possíveis relações de equivalência serão investigadas. Por fim, serão apresentadas desigualdades válidas para algumas das formulações propostas neste trabalho, para o PCD2CM.

Antes de descrevermos as formulações proposta, iremos apresentar um breve resumo contendo as informações teóricas básicas acerca de formulações de Programação Matemática.

Dado um Problema de Programação Linear,

$$\text{Min } z = cx$$

$$Ax \leq b$$

$$x \geq 0,$$

$z = cx$ é a função objetivo composta pelo vetor linha n-dimensional de custo c e o vetor coluna n-dimensional x contendo as variáveis de decisão do problema. As restrições do problema são representadas por $Ax \leq b$, sendo A uma matriz $m \times n$ contendo os coeficientes $a_{ij} \in \mathbb{Q}$ da variável de decisão j na restrição i e b o vetor coluna m-dimensional contendo o valor do-lado-direito das m restrições do problema.

Se ao Problema de Programação Linear acima for adicionada a restrição de que todas as variáveis de decisão devam ser inteiras, o problema resultante será um Problema de Programação Inteira.

$$\text{Min } z = cx$$

$$Ax \leq b$$

$$x \geq 0 \, x \in \mathbb{Z}^n.$$

Contudo, se a apenas algumas das variáveis for imposta a restrição de integralidade, o problema considerado será um Problema de Programação Inteira Mista.

$$\text{Min } z = cx + hy$$

$$Ax + Gy \leq b$$

$$x, y \geq 0 \, y \in \mathbb{Z}^p.$$

Neste caso, $G \in \mathbb{Q}^{m \times p}$ é uma matriz com dimensão $m \times p$ e y é o vetor coluna de variáveis de decisão inteiras p-dimensional.

Sem perda de generalidade, apresentaremos os resultados teóricos assumindo que os problemas considerados sejam Problemas de Programação Inteira e que encontram-se na forma de minimização. No entanto, os resultados são extensíveis para Problemas de Programação Inteira Mista (WOLSEY, 1998).

Para que um algoritmo de solução obtenha soluções exatas para um dado problema, é essencial obter uma formulação matemática para o mesmo. Em relação a isso, considere as seguintes definições (WOLSEY, 1998).

Definição 2.0.1 *O conjunto de $\mathbb{R}^n$ descrito por um conjunto de desigualdades lineares $\mathcal{R} = \{x \in \mathbb{R}^n \, : \, Ax \leq b, \, x \geq 0\}$ é uma região poliedral.*

Definição 2.0.2 (Formulação) *Uma região poliedral $\mathcal{R} \subseteq \mathbb{R}^n$ é uma formulação para o conjunto $X = \{x_1, x_2, \ldots, x_t\} \subseteq \mathbb{Z}^n$ das soluções viáveis para o Problema de Programação Inteira, se e somente se $X = \mathcal{R} \cap \mathbb{Z}^n$.*

Considere os seguintes resultados.

Definição 2.0.3 *Dado um conjunto $X = \{x_1, x_2, \ldots, x_t\} \subseteq \mathbb{R}^n$, a envoltória convexa de X, denotada por $conv(X)$, é definida como: $conv(X) = \{x \, : \, x = \sum_{i=1}^{t} \lambda_i x_i, \sum_{i=1}^{t} \lambda_i = 1, \lambda_i \geq 0$ para todo $i = 1, .., t.\}$.*

Proposição 2.0.1 *$conv(X)$ é uma região poliedral.*

Proposição 2.0.2 *Todos os pontos extremos de $conv(X)$ pertencem à X.*

Com base nos resultados apresentados acima, a seguinte relação é válida para um Problema de Programação Inteira: $\mathcal{R} \cap \mathbb{Z}^n \subseteq conv(\mathcal{R} \cap \mathbb{Z}^n)$.

Dessa forma, podemos então substituir o Problema de Programação Inteira

$$Min\{z = cx \ : \ x \in \mathcal{R} \cap \mathbb{Z}^n\}$$

pelo Problema de Programação Linear

$$Min\{z = cx \ : \ x \in conv(\mathcal{R} \cap \mathbb{Z}^n)\}.$$

Como todo ponto extremo de $conv(\mathcal{R} \cap \mathbb{Z}^n)$ pertence à $\mathcal{R} \cap \mathbb{Z}^n$ o problema seria facilmente resolvido utilizando o Método Simplex. No entanto, a menos que se prove que $P = NP$, seria impraticável, no caso geral, trabalhar com uma tal representação do problema.

A partir da relação $\mathcal{R} \cap \mathbb{Z}^n \subseteq conv(\mathcal{R} \cap \mathbb{Z}^n) \subseteq \mathcal{R}$, se torna pertinente buscar formulações $\mathcal{R}'$, tal que $\mathcal{R}'$ esteja mais próxima, de um ponto de vista geométrico, da região poliedral $conv(\mathcal{R} \cap \mathbb{Z}^n)$ que a formulação $\mathcal{R}$, ou seja, $conv(\mathcal{R} \cap \mathbb{Z}^n) \subseteq \mathcal{R}' \subset \mathcal{R}$. Com isso, a qualidade de uma formulação em relação a outra é definida a seguir.

Definição 2.0.4 *Dado um Problema de Programação Inteira e duas formulações $\mathcal{R}_1$ e $\mathcal{R}_2$ para esse problema, $\mathcal{R}_1$ é melhor que $\mathcal{R}_2$ se $\mathcal{R}_1 \subset \mathcal{R}_2$.*

Basicamente, métodos de solução para Problemas de Programação Inteira, consistem em encontrar um limitante inferior e um limitante superior para o valor da solução ótima do problema de tal forma que $\underline{z} = \bar{z}$ resulte.

O limitante inferior, para o caso de minimização, também chamado de limitante dual, está associado à uma relaxação para o problema tal como definida a seguir.

Definição 2.0.5 (GEOFFRION (2010)) *Dado um Problema de Programação Inteira, o problema*

$$\underline{z} = Min\{z = dx \ : \ x \in F\}, \tag{2.1}$$

onde $d \in \mathbb{R}$ e $F \subseteq \mathbb{R}^n_+$, é denominado uma relaxação para o Problema de Programação Inteira se e somente se:

- $\mathcal{R} \subseteq F$
- $d^t x \leq cx, \forall x \in \mathcal{R}$.

Com isso, $\underline{z}$ é um limitante inferior para o valor da função objetivo z associada à solução ótima do Problema de Programação Inteira.

Decorre diretamente dessa definição que o Problema de Programação Linear resultante da remoção da restrição de integralidade das variáveis inteiras, ou seja,

$$\text{Min}\{z = cx \ : \ Ax \leq b, x \geq 0\},$$

é uma relaxação válida para

$$\text{Min}\{z = cx \ : \ Ax \leq b, \ x \geq 0, \ x \in \mathbb{Z}^n\}.$$

A essa relaxação é dado o nome de relaxação linear.

Por outro lado, o limitante superior $\bar{z}$, para o caso de minimização, também chamado de limitante primal, corresponde ao custo de uma solução viável conhecida para o problema. Em termos práticos, essas soluções são obtidas através de procedimentos específicos denominados heurísticos. Estes são construídos de forma a encontrar soluções viáveis de "boa qualidade", em tempos computacionais aceitáveis. No entanto, não oferecem a garantia de que a solução encontrada seja ótima. Neste trabalho de tese, procedimentos heurísticos serão apresentados em detalhes no decorrer do texto.

A não ser que $conv(\mathcal{R} \cap \mathbb{Z}^n) = \mathcal{R}$, a simples resolução da relaxação linear não garante que o valor do limitante dual $\underline{z}$ seja igual ao valor de z para a solução ótima do problema.

A definição de uma formulação de boa qualidade está intimamente relacionada à definição de uma relaxação linear de boa qualidade da seguinte maneira.

Proposição 2.0.3 (WOLSEY (1998)) *Suponha que $\mathcal{R}_1$ e $\mathcal{R}_2$ sejam duas formulações para um Problema de Programação Inteira com $\mathcal{R}_1$ sendo uma formulação de melhor qualidade que $\mathcal{R}_2$, ou seja, $\mathcal{R}_1 \subset \mathcal{R}_2$. Se $\underline{z}_i = \ Min\{z = cx \ : \ x \in \mathcal{R}_i\}$ para $i = \{1, 2\}$ são os valores das relaxações lineares, então $\underline{z}_2 \leq \underline{z}_1$ para todo c.*

Com isso, um grande esforço deve ser dedicado à encontrar formulações de boa qualidade, visando facilitar o processo de obtenção da solução ótima para o problema.

Neste sentido, certas técnicas podem ser utilizadas para restringir uma formulação de forma a aproximá-la, na vizinhança da solução ótima, da região poliedral $conv(\mathcal{R} \cap \mathbb{Z}^n)$.

Definição 2.0.6 (Desigualdade válida (WOLSEY, 1998)) *Uma desigualdade $D = \{x \in \mathbb{R}^n_+ \ : \ \pi x \leq \pi_0\}$ é válida para um Problema de Programação Inteira, se $\pi x \leq \pi_0$ para qualquer solução viável desse problema. Uma classe de desigualdades é dita válida para o problema se cada desigualdade nesse classe for válida para o mesmo.*

Seja $\bar{x}$, a solução ótima da relaxação linear para o problema de Programação Inteira, se $\bar{x}$ for inteiro, então $\underline{z} = \bar{z}$ e $\bar{x}$ é uma solução ótima para o problema. Caso

contrário, o Método de Planos de Corte (veja DANTZIG *et al.* (1954) e GOMORY (1958)) identifica uma desigualdade válida que corta a solução $\bar{x}$, ou seja, $\pi\bar{x} > \pi_0$, sem que as soluções pertencentes à $\mathcal{R} \cap \mathbb{Z}^n$ sejam violadas por essa desigualdade. À tal desigualdade é dado o nome de corte. Após serem identificados, esses cortes são adicionados, na forma de restrições, à formulação $\mathcal{R}$, resultando em uma formulação $\mathcal{R}' \subset \mathcal{R}$, cujo valor de $\underline{z}' = \text{Min}\ \{z = cx\ :\ x \in \mathcal{R} \cap \{\pi\bar{x} \leq \pi_0\}\}$ é tal que $\underline{z}' \geq \underline{z}$.

2.1 Notações e Resultados Básicos

Antes de apresentar as formulações, serão introduzidas algumas notações utilizadas ao longo desta tese.

Seja $G = (V, E)$ um grafo não-direcionado tal como definido anteriormente. Uma aresta pertencente à E ligando vértices (extremidades) $i, j \in V$ é denotada por $\{i, j\}$. Por sua vez, para qualquer subconjunto $S \subset V$, o conjunto das arestas de G com uma extremidade em S e a outra em $V \setminus S$ é denotado por $\delta(S) = \{\{i, j\} \in E\ :\ i \in S\ ,\ j \in V \setminus S\}$. Com o objetivo de simplificar a notação, quando S possuir apenas um único vértice, $S = \{i\}$ por exemplo, escreveremos $\delta(i)$ ao invés de $\delta(\{i\})$. No decorrer deste texto, $\delta(S)$ será frequentemente aludido como um corte (*cutset*) de S. Além disso, como cortes estão associados à diferentes grafos neste texto, $\delta(S)$ pode eventualmente aparecer com um termo subscrito, como em $\delta_G(S)$, indicando explicitamente para qual grafo está sendo definido. $E(S) = \{\{i, j\} \in E\ :\ i, j \in S\}$ é usado para definir o subconjunto das arestas de G com ambas as extremidades em S enquanto $G_S = (S, E(S))$ representa o subgrafo de G induzido por S. Por fim, $G_{\{V \setminus Z\}} = (V \setminus Z, E(V \setminus Z))$ denota o grafo resultante da remoção de um conjunto não-vazio $Z \subset V$, do grafo $G = (V, E)$.

Para um dado subconjunto $S \subseteq V$, $\Gamma_S \subseteq V$ denota a vizinhança fechada de S, ou seja, o subconjunto de vértices de V constituído por S e todos os vértices pertencentes à $V \setminus S$ que compartilham ao menos uma aresta com vértices de S. Simplificamos a notação escrevendo Γ_i ao invés de $\Gamma_{\{i\}}$, quando S contiver apenas um único vértice i. Note também que quando a igualdade $\Gamma_S = V$ for satisfeita, S define um conjunto dominante de G.

Ao longo do texto, $D = (V, A)$ denota o grafo direcionado obtido de $G = (V, E)$ após considerar, para cada $e = \{i, j\} \in E$, dois arcos (i, j) e (j, i), com as mesma extremidades e direções opostas. A Figura 2.1 apresenta um grafo direcionado $D = (V, A)$ obtido a partir de $G = (V, E)$. Dado um subconjunto $S \subset V$, o conjunto de arcos apontando de um vértice de S para um vértice de $V \setminus S$ é denotado por $\delta^+(S)$. Por conseguinte, $\delta^-(S)$ identifica o conjunto de arcos apontando de vértices de $V \setminus S$ para vértices de S. Devido à orientação de seus arcos, $\delta^+(S)$ e $\delta^-(S)$ serão chamados respectivamente de cortes para frente (*forward*) e para trás (*backward*).

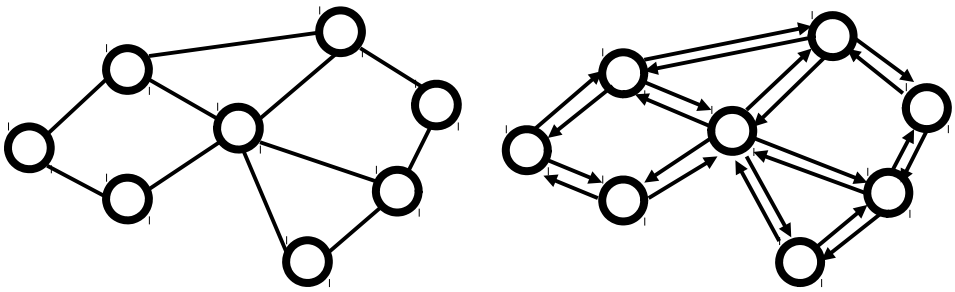

Figura 2.1: Um grafo $G = (V, E)$ e o seu correspondente grafo direcionado $D = (V, A)$.

Visando simplificar a notação, quando S contiver apenas um único vértice, $S = \{i\}$ por exemplo, $\delta^+(i)$ e $\delta^-(i)$ serão utilizados no lugar de $\delta^+(\{i\})$ e $\delta^-(\{i\})$, respectivamente. Um conjunto dominante $W \subseteq V$ é 2-aresta (respectivamente 2-vértice) conexo para G se, para cada par de vértices distintos $u, v \in W$, existirem dois caminhos aresta (respectivamente vértice) disjuntos em G_W conectando u e v. Deixando de lado o tipo de 2-conexidade envolvida, denotamos genericamente por PCD2CM o problema de encontrar um conjunto W de cardinalidade mínima que seja 2-aresta ou 2-vértice conexo.

Para dois vértices distintos quaisquer $s, t \in V$, qualquer corte em G com $s \in S$ e $t \in V \setminus S$ é chamado de corte $s - t$. Para um grafo direcionado D, com s e t como anteriormente definidos, fazemos uma distinção. Um corte s-t corresponde aos arcos em $\delta^+(S)$ enquanto um corte t-s corresponde àqueles em $\delta^-(S)$.

O teorema a seguir apresenta um resultado amplamente utilizado na construção de modelos para problemas de desenho de redes (GRÖTSCHEL *et al.*, 1995; MAGNANTI e RAGHAVAN, 2005), nos quais alguns requisitos específicos de conexidade são impostos.

Teorema 2.1.1 (Teorema de Menger, como descrito em DIESTEL (2010)) *Seja $G = (V, E)$ um grafo não-direcionado e $i, j \in V$ um par de vértices distintos quaisquer de G. A quantidade mínima de arestas em qualquer corte separando i e j é igual à quantidade máxima de caminhos aresta-disjuntos conectando i e j.*

Uma orientação de um grafo $G = (V, E)$ não-direcionado é definida como um digrafo $D' = (V, A')$ tal que, para cada aresta $\{i, j\} \in E$, existe apenas um único arco $(i, j) \in A'$ ou $(j, i) \in A'$. Assim sendo, podemos caracterizar um grafo 2-aresta conexo em termos de uma orientação conforme o resultado apresentado a seguir.

Teorema 2.1.2 (ROBBINS (1939)) *Um grafo $G = (V, E)$ é 2-aresta conexo se e somente se existir uma orientação D' de G tal que, para cada par de vértices distintos $u, v \in V$, tivermos um par de caminhos arco disjuntos em D' conectando u à v e vice-versa.*

Uma versão do Teorema 2.1.2 para 2-vértice conexidade é expressa pelo teorema descrito a seguir.

Teorema 2.1.3 (CHIMANI *et al.* (2010)) *Um grafo $G = (V, E)$ é 2-vértice conexo se e somente se para um vértice $t \in V$ escolhido arbitrariamente, existir uma orientação $D' = (V, A')$ tal que apenas um arco aponta para o vértice t e para todo vértice $v \in V \setminus \{t\}$, existe um par de caminhos direcionados em D' indo de t até v e de v até t, ambos possuindo vértices distintos exceto por t e v.*

Como iremos trabalhar com subgrafos 2-{aresta,vértice} conexos, o resultado imediatamente a seguir estabelece uma condição para que um grafo seja 2-vértice conexo e se baseia na contração de grafos. O resultado imediatamente subsequente identifica a existência de uma relação de transitividade entre qualquer par de caminhos 2-aresta-disjuntos com um vértice em comum.

Teorema 2.1.4 (DIESTEL (2010)) *Um grafo é 2-vértice conexo se e somente se pode ser obtido através da expansão de qualquer de seus subgrafos 2-vértice conexos iterativamente adicionando ao subgrafo (e suas subsequentes expansões) caminhos com ambas extremidades pertencentes ao subgrafo corrente.*

Lema 2.1.1 *Seja $G = (V, E)$ um grafo não-direcionado e i,j e k três vértices distintos pertencentes a este grafo. Se existirem dois caminhos aresta-disjuntos conectando i e j e dois caminhos aresta-disjuntos conectando j e k, então existem dois caminhos aresta-disjuntos conectando i e k.*

Prova 2.1.1 *Para chegar a uma contradição, vamos supor que não existam dois caminhos aresta-disjuntos conectando os vértices i e k em G. Com isso, pelo Teorema de Menger existe um conjunto S, tal que $i \in S$, $k \in V \setminus S$ e $|\delta(S)| \leq 1$. Dessa forma, para que existam dois caminhos aresta-disjuntos conectando i e j, j deve pertencer à S, e de maneira análoga, para que existam dois caminhos aresta-disjuntos conectando k e j, j deve também pertencer à $V \setminus S$. Como $S \cap (V \setminus S) = \emptyset$, chegamos a uma contradição para a suposição inicial de que não existem dois caminhos aresta-disjuntos conectando i e j.* □

Finalmente, fechando esta subseção, o resultado a seguir estabelece uma condição suficiente para que um grafo possua um solução para a variante 2-aresta conexa do PCD2CM.

Teorema 2.1.5 *Seja $G = (V, E)$ um grafo conexo e $L = \{l \in V : |\delta(l)| = 1\}$ seu conjunto de folhas. $V \setminus L$ é um conjunto dominante 2-aresta conexo de G se e somente se existir um subconjunto $W \subseteq V \setminus L$ que seja dominante 2-aresta conexo para G.*

Prova 2.1.2 *Se $V \setminus L$ é um conjunto dominante 2-aresta conexo de G, o resultado é obtido diretamente ao se tomar $W = V \setminus L$. Consequentemente, deve-se provar que se existir um conjunto dominante 2-aresta conexo $W \subseteq V \setminus L$, então o conjunto $V \setminus L$ é dominante 2-aresta de G. Se W é um conjunto dominante 2-aresta para G, $V \setminus W$ é dominado por W e como G_W é 2-aresta conexo, $L \subseteq (V \setminus W)$. Se $L = V \setminus W$ então $W = V \setminus L$ e portanto $V \setminus L$ é um conjunto dominante 2-aresta conexo para G. Se $L \neq V \setminus W$, temos que $V \setminus W \not\subset L$ e consequentemente deve existir um conjunto $V' \subset V$ tal que $V' \cap L = \emptyset$ e $V' \subset (V \setminus W)$. Lembrando que, para cada $i \in V'$, $|\delta(i)| \geq 2$ é válido, se $|\Gamma_i \cap W| \geq 2$, então $\{i\} \cup W$ é 2-aresta conexo. Por outro lado, se $|\Gamma_i \cap W| = 1$, deve então existir um $j \in V'$ tal que $j \in \Gamma_i$. Dessa forma, $\{i, j\} \cup W$ deverá ser 2-aresta conexo e $V' \cup W$ um conjunto dominante 2-vértice conexo para G. Como $V' \cap L = \emptyset$ e $V' \subset (V \setminus W)$, $V' \cup W$ deverá ser igual a $V \setminus L$. Portanto, pode-se concluir que $V \setminus L$ é um conjunto dominante 2-aresta conexo para G.* $\square$

Embora o Teorema 2.1.5 apresente condições suficientes para que um grafo G contenha conjuntos dominantes 2-aresta conexos, o mesmo argumento não pode ser estendido à variante 2-vértice conexa do problema. Tal fato pode ser constatado no exemplo descrito na Figura 2.2, onde existe um subgrafo dominante 2-vértice conexo de G e $V \setminus L$ induz um subgrafo 2-aresta conexo. No entanto, se $V \setminus L$ induz um subgrafo 2-vértice conexo, ainda existe um conjunto dominante $W \subseteq V \setminus L$ tal que G_W é 2-vértice conexo, já que W pode ser igual à $V \setminus L$.

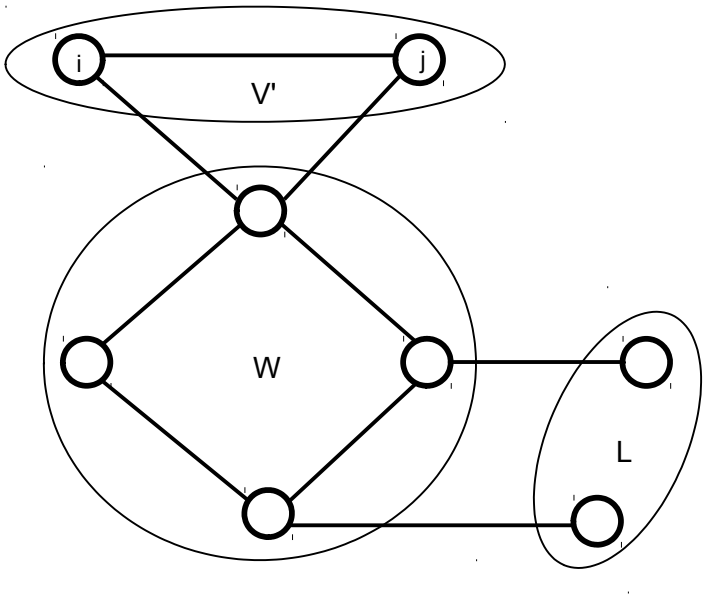

Figura 2.2: W induz um subgrafo 2-vértice conexo enquanto $V \setminus L$ induz um subgrafo 2-aresta conexo.

2.2 Formulações Para a Variante 2-Aresta Conexa

Nesta sessão vamos introduzir quatro formulações para a variante 2-aresta conexa do PCD2CM. Uma formulação é definida diretamente sobre o grafo $G = (V, E)$, outra se utiliza de multi-fluxos e, finalmente, as duas restantes são reformulações definidas em grafos direcionados.

2.2.1 Uma Formulação Definida Sobre o Espaço Natural de Variáveis

Esta formulação se origina diretamente do Teorema de Menger e é aplicada à variante 2-aresta conexa do problema. Ela envolve dois conjuntos de variáveis, $\{y_i \in \{0,1\} : i \in V\}$ e $\{0 \leq x_e \leq 1 : e \in E\}$. O primeiro conjunto é responsável por identificar os vértices pertencentes ao conjunto dominante W. Assim sendo, $y_i = 1$ implica na pertinência do vértice i ao conjunto W, com $y_i = 0$ implicando o contrário. O segundo conjunto de variáveis traz informação similar com relação às arestas de $G_W = (W, E(W))$, ou seja, o subgrafo de G induzido por W. Dessa forma, $x_e = 1$ resulta se e pertence ao conjunto G_W enquanto $x_e = 0$ se aplica em caso contrário. Em conjunto com essas variáveis, considere a região poliedral $\mathcal{R}_1$ definida como

$$\sum_{j \in \Gamma_i \setminus \{i\}} y_j \geq y_i + 1, \qquad i \in V \tag{2.2}$$

$$x_e \leq y_k, \;\; e = \{i, j\} \in E, \;\; k \in \{i, j\} \tag{2.3}$$

$$\sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1), \qquad S \subset V, \quad i \in S, \quad j \in V \setminus S \tag{2.4}$$

$$0 \leq x_e \leq 1, \quad e \in E \tag{2.5}$$

$$0 \leq y_i \leq 1, \quad i \in V. \tag{2.6}$$

Uma formulação para o PCD2CM é então dada por

$$\min \{\sum_{i \in V} y_i : (\mathbf{x}, \mathbf{y}) \in \mathcal{R}_1 \cap (\mathbb{R}_+^{|E|}, \mathbb{B}^{|V|})\} \tag{2.7}$$

e sua relaxação linear correspondente

$$\min \{\sum_{i \in V} y_i : (\mathbf{x}, \mathbf{y}) \in \mathcal{R}_1\}. \tag{2.8}$$

As desigualdades (2.2) impõem a condição de dominância. Desse modo, se $i \in V$ pertence ao conjunto dominante W, ao menos dois vértices vizinhos de i devem também pertencer à W. Por outro lado, se i não pertence à W, ao menos um vértice vizinho de i deve pertencer a esse conjunto. Desigualdades (2.3) indicam que uma aresta $e \in E$ só pode pertencer à G_W quando ambos os seus vértices terminais pertencerem à W. As desigualdades (2.4) garantem que o subgrafo G_W é 2-aresta conexo. Portanto, se dois vértices de G, i e j por exemplo, pertencem à W, qualquer corte de G_W separando i e j deve conter ao menos duas arestas como especificado pelo Teorema de Menger. Por fim, as desigualdades (2.5) e (2.6) impõem os limitantes necessários nas variáveis da formulação.

2.2.2 Uma Formulação Multi-Fluxos

Em seguida, apresentaremos uma formulação compacta para a variante 2-aresta conexa do PCD2CM. Esta conta com um quantidade polinomial de restrições com relação a dimensão do grafo. O apelo para este tipo de formulação origina-se do fato de que, para instâncias de dimensão pequena, esta pode, a principio, ser resolvida diretamente por um *software* genérico de Programação Inteira Mista.

A formulação é definida simultaneamente sobre o grafo não-direcionado $G = (V, E)$ e seu correspondente direcionado, $D = (V, A)$. Soluções viáveis para esta formulação, além do conjunto dominante W, também caracterizam os subgrafos correspondentes $G_W = (W, E(W))$ e $D_W = (W, A(W))$, respectivamente de G e D. Sendo D_W o subgrafo induzido pelo conjunto dominante W em D. A ideia básica associada é utilizar fluxos em rede para impor a conectividade desejada para a solução. Esse objetivo é alcançado impondo, para todo par de vértices distintos $s, t \in W$,

com $s < t$, a existência em D_W de dois caminhos distintos em termos de seus arcos, conectando s à t. A formulação origina-se da formulação introduzida em MACULAN (1986), para o Problema da Arborescência Geradora Mínima. Esta envolve, dentre outras, as variáveis previamente definidas, $\mathbf{x}$ e $\mathbf{y}$. Além disso, também conta com variáveis $\{0 \leq f_{ij}^{st} \leq 1 : s, t \in V,\, s < t,\, (i,j) \in A\}$, onde f_{ij}^{st} assume um valor igual a 1 se existir um caminho em D_W, com origem em s e destino em t, contendo o arco $(i,j) \in A$. Caso contrário, $f_{ij}^{st} = 0$ se aplica. Finalmente, para garantir que pelo menos dois caminhos arco-disjuntos existem em D_W, duas unidades de fluxo serão enviadas de s para t, através de uma rede em que a capacidade dos arcos é igual a 1. Tal condição é modelada através de variáveis $\{0 \leq q_{ij} \leq 1 : i, j \in V,\, i < j\}$, com $q_{ij} = 1$ indicando que $i, j \in W$ e $q_{ij} = 0$ resultando em caso contrário.

Para as variáveis definidas acima, considere a região poliedral $\mathcal{R}_2$ descrita como

$$\sum_{j \in V,(jl) \in A} f_{jl}^{st} - \sum_{j \in V,(lj) \in A} f_{lj}^{st} = \begin{cases} -2q_{st} \text{ se } l = s \\ 2q_{st} \text{ se } l = t \\ 0 \text{ caso contrário} \end{cases}, \quad \forall\, s, t, l \in V,\ s < t \tag{2.9}$$

$$f_{ij}^{st} + f_{ji}^{st} \leq x_e, \quad \forall\ e = \{i,j\} \in E \text{ e } s, t \in V,\ s < t \tag{2.10}$$

$$x_e \leq y_k,\ \ e = \{i,j\} \in E,\ \ k \in \{i,j\} \tag{2.11}$$

$$q_{ij} \geq y_i + y_j - 1,\ \ i, j \in V,\ i < j \tag{2.12}$$

$$q_{ij} \leq y_i \text{ e } q_{ij} \leq y_j,\ \ i, j \in V,\ i < j \tag{2.13}$$

$$\sum_{j \in \Gamma_i \setminus \{i\}} y_j \geq y_i + 1,\ \ i \in V \tag{2.14}$$

$$0 \leq x_e \leq 1,\ \ e \in E \tag{2.15}$$

$$0 \leq y_i \leq 1,\ \ i \in V \tag{2.16}$$

$$0 \leq q_{ij} \leq 1,\ \ i, j \in V,\ i < j \tag{2.17}$$

$$0 \leq f_{ij}^{st} \leq 1,\ \ (i,j) \in A\ \ s, t \in V,\ s < t. \tag{2.18}$$

Uma formulação multi-fluxo para a variante 2-aresta conexa do PCD2CM é então dada por

$$\min \{\sum_{i \in V} y_i : (\mathbf{f}, \mathbf{q}, \mathbf{x}, \mathbf{y}) \in \mathcal{R}_2 \cap (\mathbb{R}_+^{\frac{|V|.(|V|-1).|A|}{2}}, \mathbb{R}_+^{\frac{|V|(|V|-1)}{2}}, \mathbb{R}_+^{|E|}, \mathbb{Z}^{|V|})\} \tag{2.19}$$

e sua relaxação linear correspondente é definida por

$$\min \{\sum_{i \in V} y_i : (\mathbf{f}, \mathbf{q}, \mathbf{x}, \mathbf{y}) \in \mathcal{R}_2\}. \tag{2.20}$$

As desigualdades (2.10) e (2.11) impõem, para qualquer par de vértices distintos $s, t \in W$, com $s < t$, que f_{ij}^{st} e f_{ji}^{st} podem apenas assumir valores não-nulos se $e = \{i, j\} \in E$ pertencer ao subgrafo G_W. Além disso, para s e t já definidos, desigualdades (2.9), (2.12), (2.13), (2.17), e (2.18) impõem que duas unidades de um único tipo de mercadoria (*commodity*) são enviadas de s para t. Por fim, as desigualdades (2.14), como anteriormente impõem a condição de dominância.

A formulação (2.19) é mais compacta que a formulação (2.7) em termos do número de restrições que utiliza. Por outro lado, envolve uma quantidade enorme de variáveis e restrições. A formulação (2.19) contém $\frac{|V|^3+(|V|^2(|E|+2))-(|V|(|E|+1))+4|E|}{2}$ restrições e $|E|(|V|^2 - |V| + 2) + \frac{|V|^2}{2}$ variáveis. Por exemplo, para um grafo com 100 vértices e 1000 arestas, obtemos 5.461.950 restrições e 9.907.000 variáveis, o que é excessivo para se considerar diretamente em um *software* de Programação Inteira Mista.

2.2.3 Uma Reformulação Usando Grafos Direcionados.

Uma motivação para introduzirmos uma reformulação de (2.7) baseada em grafos direcionados é que, tipicamente, para problemas de desenho de redes tal como o que discutimos neste trabalho de tese, as relaxações lineares tendem a gerar limitantes duais de melhor qualidade (MAGNANTI e RAGHAVAN, 2005). A formulação a ser descrita a seguir resulta diretamente do digrafo $D = (V, A)$ e envolve, além das variáveis $\mathbf{y}$ previamente definidas, variáveis $\{0 \leq h_{ij} \leq 1 : (i, j) \in A\}$, correspondendo aos arcos de D. Esta reformulação é obtida substituindo x_e por $(h_{ij} + h_{ji})$ em (2.7), assumindo que $e = \{i, j\} \in E$. No entanto, como já definido, tal reformulação deve conter mais soluções viáveis que (2.7). Tome, por exemplo, os três subgrafos distintos de D e o único subgrafo de G representados na Figura 2.3. Embora os subgrafos direcionados definam soluções viáveis para a reformulação, de acordo com o Teorema (2.1.2), somente o último deles seria válido. Ademais, este seria também o único correspondendo à solução viável para o PCD2CM definida pelo subgrafo não-direcionado da figura. De acordo com o teorema, se um grafo não-direcionado G_W é 2-aresta conexo, este pode ser mapeado diretamente em um grafo direcionado D_W que contem um único arco para cada aresta de G_W. Além disso, D_W é tal que, para cada par de vértices $i, j \in W$ distintos, dois caminhos arco-disjuntos devem existir em D_W, respectivamente indo de i para j e de j para i.

Tendo em mente as observações feitas acima, uma reformulação direcionada de (2.7) é definida através da seguinte região poliedral $\mathcal{R}_3$

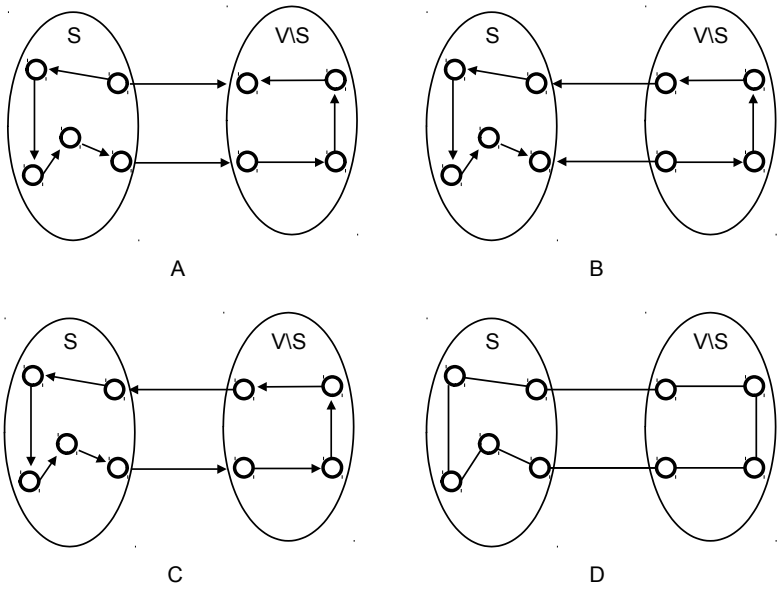

Figura 2.3: Apenas o corte em C é suficiente para garantir 2-aresta conectividade.

$$\sum_{a \in \delta^+(S)} h_a \geq y_i + y_j - 1, \;\; S \subset V, \;\; i \in S, \; j \in V \setminus S \tag{2.21}$$

$$\sum_{j \in \Gamma_i \setminus \{i\}} y_j \geq y_i + 1, \;\; i \in V \tag{2.22}$$

$$h_{ij} + h_{ji} \leq y_i, \;\; (i,j) \in A \tag{2.23}$$

$$0 \leq h_{ij} \leq 1, \;\; (i,j) \in A \tag{2.24}$$

$$0 \leq y_i \leq 1, \;\; i \in V. \tag{2.25}$$

Por conseguinte, uma reformulação direcionada para o PCD2CM é dada por

$$\min \{\sum_{i \in V} y_i : (\mathbf{h}, \mathbf{y}) \in \mathcal{R}_3 \cap (\mathbb{R}_+^{|A|}, \mathbb{B}^{|V|})\}, \tag{2.26}$$

com uma relaxação linear correspondente,

$$\min \{\sum_{i \in V} y_i : (\mathbf{h}, \mathbf{y}) \in \mathcal{R}_3\}. \tag{2.27}$$

Desigualdades (2.21) garantem, para qualquer par de vértices $i, j \in V$ distintos, que qualquer corte separando i de j contém ao menos $(y_i + y_j - 1)$ arcos. Dessa forma, como sugerido pelo Teorema 2.1.2, devem envolver ao menos um arco, quando ambos os vértices pertencerem ao conjunto dominante W, ou seja, quando $y_i = y_j = 1$. Caso contrário, as desigualdades serão inativas. Desigualdades (2.23) impõem que

existe apenas um arco em D_W correspondendo a uma aresta de G_W. Por fim, como anteriormente, desigualdades (2.22) impõem a condição de dominância.

2.2.4 Reformulação Direcionada Usando Um Vértice Artificial

Pelo Lema (2.1.1) podemos inferir, para um grafo conexo $G = (V, E)$, que se existir um vértice $i \in V$ tal que, para cada $j \in V \setminus \{i\}$, existem dois caminhos aresta-disjuntos conectando i a j, então também existirão dois caminhos aresta-disjuntos conectando cada par de vértices em V. Ou seja, o grafo G é 2-aresta conexo.

Essa ideia permite construir uma outra reformulação direcionada, diferente de (2.26). Nesta formulação, dado um conjunto dominante W e um vértice $i \in W$, a 2-aresta conexidade do subgrafo induzido pelo conjunto dominante W pode ser estabelecida impondo que, para todo vértice $j \in W \setminus \{i\}$, existam dois caminhos arco disjuntos saindo de i e chegando em j e vice-versa. No entanto, já que os vértices que farão parte do conjunto dominante são desconhecidos, introduzimos um grafo estendido $D_r = (V_r, A_r)$ com $V_r = V \cup \{r\}$ e $A_r = A \cup \{(r, i) : i \in V\}$, onde r é um vértice raiz artificial. O grafo resultante é mostrado na Figura 2.4. O objetivo de se utilizar o vértice artificial é selecionar um vértice i pertencente à W, para fazer, de fato, o papel de raiz do subgrafo de D a ser identificado.

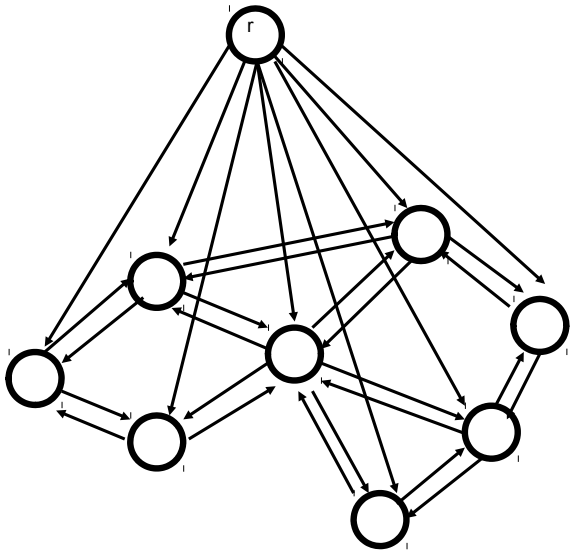

Figura 2.4: Um grafo $D_r = (V_r, A_r)$.

Pela definição de conjunto dominante, ao menos um vértice pertencente ao conjunto Γ_i pertencerá ao conjunto dominante, ou seja, $\Gamma_i \cap W \neq \emptyset$. Dessa forma, é necessário apenas impor que o vértice raiz artificial esteja conectado à apenas um vértice do conjunto Γ_i. Seja $\Gamma_{min} = argmin\{|\Gamma_i| : i \in V\}$ a vizi-

nhança fechada de menor cardinalidade do grafo G. Dessa maneira, o grafo estendido $D_r = (V_r, A_r)$ é redefinido utilizando apenas o conjunto reduzido de arcos $A_r = A \cup \{(r, i) : i \in \Gamma_{min}\}$. O subgrafo induzido em D_r por uma solução viável define uma topologia de solução que deve conter um conjunto dominante $W \subseteq V$ de D, o nó raiz artificial r e um subconjunto particular de arcos de A_r. Este subconjunto deve contar com apenas um arco apontando de r para um único vértice i de $\Gamma_{min} \cap W$. Além desse arco, o subconjunto deve conter, para cada outro vértice $j \in W$, caminhos arco-disjuntos conectando i à j e vice-versa. A Figura 2.5 ilustra uma topologia associada à uma solução viável para o grafo D_r representado na Figura 2.4. Nesta figura, as linhas pontilhadas representam arcos ou vértices que fazem parte do grafo D_r e não pertencem à topologia de solução.

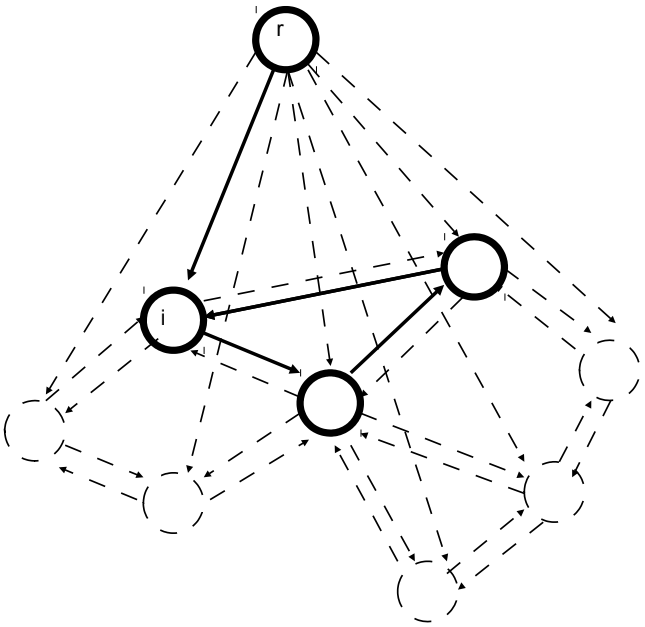

Figura 2.5: Topologia de solução viável para a reformulação direcionada com vértice artificial.

A formulação descrita acima envolve variáveis $\{y_i \in \{0, 1\} : i \in V\}$, definidas como anteriormente. Envolve também variáveis $\{0 \leq h_{(i,j)} \leq 1 : (i, j) \in A_r \text{ e } i \neq r\}$ para determinar os arcos do subgrafo 2-arco conexo induzido por W. Utiliza ainda variáveis $\{z_i \in \{0, 1\} : (r, i) \in A_r\}$ para determinar o arco único que sai de r e aponta para um vértice de W. É definida pela região poliedral $\mathcal{R}_4$, descrita como

$$\sum_{a \in \delta^-(S)} h_a + \sum_{j \in \Gamma_{min} \setminus S} z_j \geq y_i, \qquad S \subset V_r, \quad r \in S, \quad i \in V \setminus S \tag{2.28}$$

$$\sum_{a \in \delta^+(S)} h_a + \sum_{j \in \Gamma_{min} \setminus S} z_j \geq y_i, \qquad S \subset V_r, \quad r \in S, \quad i \in V \setminus S \tag{2.29}$$

$$\sum_{i \in \Gamma_{min}} z_i = 1 \tag{2.30}$$

$$z_i \leq y_i, \qquad i \in \Gamma_{min} \tag{2.31}$$

$$\sum_{j \in \Gamma_i \setminus \{i\}} y_j \geq y_i + 1, \qquad i \in V \tag{2.32}$$

$$h_{(i,j)} + h_{(j,i)} \leq y_i \qquad (i,j) \in A, \quad i,j \neq r \tag{2.33}$$

$$0 \leq h_a \leq 1, \quad a \in A \tag{2.34}$$

$$0 \leq z_i \leq 1, \quad i \in \Gamma_{min} \tag{2.35}$$

$$0 \leq y_i \leq 1, \quad i \in V, \tag{2.36}$$

sendo dada por

$$\min \{\sum_{i \in V} y_i : (\mathbf{h}, \mathbf{z}, \mathbf{y}) \in \mathcal{R}_4 \cap (\mathbb{R}_+^{|A|}, \mathbb{B}^{|\Gamma_{min}|}, \mathbb{B}^{|V|})\} \tag{2.37}$$

e tendo uma relaxação linear

$$\min \{\sum_{i \in V} y_i : (\mathbf{h}, \mathbf{z}, \mathbf{y}) \in \mathcal{R}_4\}. \tag{2.38}$$

A existência de caminhos saindo do único vértice $i \in \Gamma_{min}$ conectado a r para um vértice $j \in W \setminus \{i\}$ e voltando do vértice j para i são respectivamente garantidos pelas desigualdades de corte (2.28) e (2.29). Além disso, as restrições (2.30) e (2.31) garantem que apenas um arco saindo de r é selecionado. Por fim, as restrições restantes já foram discutidas anteriormente para a formulação (2.26).

2.3 Formulações Para a Variante 2-Vértice Conexa

Nesta seção serão apresentadas nossas formulações para a variante 2-vértice conexa do PCD2CM.

2.3.1 Formulação Definida Sobre G

A formulação é obtida restringindo ainda mais a região poliedral $\mathcal{R}_1$, para impor 2-vértice conectividade a G_W. Desigualdades

$$\sum_{e \in \delta_{G_{\{V \setminus \{k\}\}}}(S)} x_e \geq y_i + y_j - 1 \qquad S \subset V \setminus \{k\},\ i \in S,\ j \in V \setminus S \cup \{k\}\,, k \in V, \tag{2.39}$$

são então utilizadas para essa função. Isto é feito de acordo com a investigação con-

duzida em GRÖTSCHEL *et al.* (1995), para redes sobreviventes. A ideia implícita nas mesmas é que um grafo não-direcionado $G = (V, E)$ é 2-vértice conexo se este continua conectado após a remoção de um único vértice, no caso o vértice k. Uma região poliedral $\mathcal{R}_5$ associada à formulação é definida pela interseção de $\mathcal{R}_1$ com as desigualdades (2.39). Dessa maneira, a variante 2-vértice conexa do PCD2CM é formulada como

$$\min \{\sum_{i \in V} y_i : (\mathbf{x}, \mathbf{y}) \in \mathcal{R}_5 \cap (\mathbb{R}_+^{|E|}, \mathbb{B}^{|V|})\}, \tag{2.40}$$

tendo uma relaxação linear

$$\min \{\sum_{i \in V} y_i : (\mathbf{x}, \mathbf{y}) \in \mathcal{R}_5\}. \tag{2.41}$$

2.3.2 Reformulação Direcionada

O Teorema 2.1.3 sugere uma caracterização de grafos 2-vértice conexos através de uma orientação D' do grafo G. Este teorema é similar ao Teorema 2.1.2 utilizado na Seção (2.2.3) para obter uma reformulação direcionada da formulação (2.7) para a variante 2-aresta conexa do PCD2CM. No entanto, para a variante 2-vértice conexa do problema duas condições devem ser impostas. A primeira estabelece que, dado um vértice arbitrário $t \in V$, dois caminhos vértice-disjuntos devem existir na orientação D', conectando t a cada vértice de $v \in V \setminus \{t\}$ e vice-versa. Adicionalmente, para que G seja 2-vértice conexo, é também necessário a existência de um único arco entrando no vértice t, ou seja, que $|\delta^-(t)| = 1$.

Obtemos uma reformulação direcionada para (2.40), com base na reformulação direcionada com nó raiz artificial (2.37). Dessa forma, a região de viabilidade a considerar é composta pela união da região poliedral $\mathcal{R}_4$ com um conjunto de desigualdades que garantem que uma solução viável D_W corresponde à orientação definida pelo Teorema 2.1.3. Para tal são adicionados dois conjuntos de restrições à região poliedral $\mathcal{R}_4$. O primeiro deles é definido como

$$\sum_{a \in \delta^-_{D_{\{V\setminus\{k\}\}}}(S)} h_a + \sum_{a \in \delta^+_{D_{\{V\setminus\{k\}\}}}(S)} h_a + \sum_{i \in \Gamma_{min} \setminus S} z_i \geq y_j, S \subset V \setminus \{k\}, j \in V \setminus S \cup \{k\}, k \in V, \tag{2.42}$$

sendo tais desigualdades análogas àquelas definidas por (2.39). Ou seja, para cada $j, k \in V$, se j fizer parte do conjunto dominante W, $y_j = 1$ resulta e qualquer corte $\delta^-_{D_{\{V\setminus\{k\}\}}}$ ou $\delta^+_{D_{\{V\setminus\{k\}\}}}$ no digrafo $D_{\{V\setminus\{k\}\}}$ deve conter pelo menos um arco. Já o segundo conjunto de restrições é definido como

$$\sum_{a \in \delta^-(i)} h_a \leq M_i(1 - z_i) + 1 \qquad i \in \Gamma_{min}. \tag{2.43}$$

Nas desigualdades (2.43), para todo $i \in \Gamma_{min}$, se i pertence ao conjunto dominante W e este for o único vértice conectado ao vértice raiz artificial, $z_i = 1$ resulta e a soma das variáveis h_a correspondente a todos os arcos que entram em i deve ser no máximo igual a um. Caso contrário, $z_i = 0$ resulta e tal soma deverá ser no máximo $M_i + 1$. Como o número de arcos entrando em i, no digrafo D, é igual à $|\Gamma_i|$ e ao menos um arco deve sair de i se este fizer parte do conjunto dominante, M_i pode assumir o valor $|\Gamma_i| - 2$. Dessa maneira a desigualdade (2.43) pode ser reescrita na forma mais justa

$$\sum_{a \in \delta^-(i)} h_a \leq (2 - |\Gamma_i|)z_i + |\Gamma_i| - 1, \qquad i \in \Gamma_{min}. \tag{2.44}$$

Denotamos por $\mathcal{R}_6$ a região poliedral resultante da interseção de $\mathcal{R}_4$ com as desigualdades (2.42) e (2.44). Dessa forma, uma reformulação direcionada para a variante 2-vértice conexa do PCD2CM é dada por

$$\min \{\sum_{i \in V} y_i : (\mathbf{h}, \mathbf{z}, \mathbf{y}) \in \mathcal{R}_6 \cup (\mathbb{R}_+^{|A|}, \mathbb{B}^{|\Gamma_{min}|}, \mathbb{B}^{|V|})\} \tag{2.45}$$

e sua relaxação linear é definida como

$$\min \{\sum_{i \in V} y_i : (\mathbf{h}, \mathbf{z}, \mathbf{y}) \in \mathcal{R}_6\}. \tag{2.46}$$

2.3.3 Formulação Multi-Fluxos

Como proposto anteriormente para a variante 2-aresta conexa do PCD2CM, uma formulação baseada em multi-fluxos pode também ser definida para a variante 2-vértice conexa do problema. Para isso basta garantir, para todo par de vértices $i, j \in W \subseteq V$, que os caminhos utilizados para transportar o par de mercadorias demandadas de i para j sejam, a menos de i e j, simultaneamente aresta e vértices disjuntos. Denote por $\mathcal{R}_7$ a região poliedral resultante da inserção de $\mathcal{R}_2$ com as seguintes restrições,

$$\sum_{i \in \Gamma_l, (il) \in A} f_{il}^{st} \leq 1 \;\; s, t \in V, \; s \leq t, \; l \in V \setminus \{s, t\}. \tag{2.47}$$

De acordo com essas restrições, duas unidades de fluxo não poderão mais passar por um único vértice. Assim sendo, todos os fluxos originários de um vértice s deverão percorrer caminhos vértice-disjuntos até o vértice t. Dessa maneira, para

cada par $s - t$, os fluxos obtidos identificam um ciclo no subgrafo induzido pelo conjunto dominante. Disto resulta que

$$\min \{\sum_{i \in V} y_i : (\mathbf{f}, \mathbf{x}, \mathbf{z}, \mathbf{y}) \in \mathcal{R}_7 \cap (\mathbb{R}_+^{2.|V|.|A_r|}, \mathbb{R}_+^{|E|}, \mathbb{B}^{|\Gamma_{min}|}, \mathbb{B}^{|V|})\} \tag{2.48}$$

é uma formulação multi-fluxos para a variante 2-vértice conexa do PCD2CM. Por sua vez, sua relaxação linear correspondente é definida como

$$\min \{\sum_{i \in V} y_i : (\mathbf{f}, \mathbf{x}, \mathbf{z}, \mathbf{y}) \in \mathcal{R}_7\}. \tag{2.49}$$

2.4 Desigualdades Válidas

Algumas desigualdades válidas para (2.7) e (2.40) são descritas a seguir. Para tanto, note que, pela definição de um conjunto dominante W, de $G = (V, E)$, temos que $W \cap \Gamma_i \neq \emptyset$, para qualquer $i \in V$. Logo, para qualquer par de vértices distintos $i, j \in V$ tais que $\Gamma_i \cap \Gamma_j = \emptyset$, qualquer corte separando Γ_i e Γ_j deve conter ao menos duas arestas, como pode ser observado na Figura 2.6. Desigualdades válidas para o PCD2CM, que se originam daquelas desigualdades introduzidas por SIMONETTI *et al.* (2011) para o Problema do Conjunto Dominante Conexo Mínimo, são então dadas por

$$\sum_{e \in \delta(S))} x_e \geq 2 \quad \Gamma_i \subset S, \ \Gamma_j \subset V \setminus S, \ \Gamma_i \cap \Gamma_j = \emptyset. \tag{2.50}$$

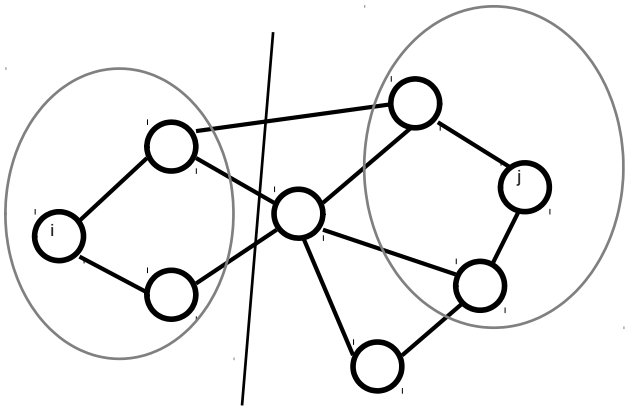

Figura 2.6: Corte separando os conjuntos Γ_i e Γ_j.

2.5 Relação de Equivalência Entre Formulações

Duas formulações para um mesmo problema são ditas equivalentes quando suas relaxações lineares definem um mesmo espaço de soluções viáveis.

Considere as duas formulações que se seguem, para um mesmo problema. A primeira é descrita por

$$\text{Min}\{cx \ : \ x \in P \cap \mathbb{Z}^n\}, \tag{2.51}$$

sendo P uma região poliedral tal que $P \subset \mathbb{R}^n$. A segunda é uma formulação "estendida", que é dada por

$$\text{Min}\{cx \ : \ (x, f) \in Q \cap (\mathbb{Z}^n, \mathbb{R}^p)\}. \tag{2.52}$$

Para determinar se há uma relação de equivalência entre as formulações (2.51) e (2.52) é necessário definir uma projeção da região poliedral Q no espaço das variáveis originais x.

Definição 2.5.1 (Projeção) *Dada uma região poliedral $Q \subseteq (\mathbb{R}^n, \mathbb{R}^p)$, a projeção de Q no subespaço $\mathbb{R}^n$ é definida como*

$$Proj_x(Q) = \{x \in \mathbb{R}^n \ : \ (x, f) \in Q, \ \textit{para algum } f \in \mathbb{R}^p\}.$$

De acordo com esta definição, a região poliedral obtida pela eliminação das variáveis f é uma projeção da região poliedral Q no espaço de variáveis originais x. Como esta projeção e a região poliedral P estão sob o mesmo espaço de variáveis, podemos então compará-las.

2.5.1 Caso 2-Aresta Conexo

Relações de dominância entre as relaxações lineares das diferentes formulações aqui introduzidas para a variante 2-aresta conexa do PCD2CM, são discutidas a seguir.

Para tanto, vamos inicialmente mostrar que a formulação (2.7) e a formulação multi-fluxos, (2.19), têm um mesmo valor ótimo para as suas respectivas relaxações lineares. Para obter esse resultado, tendo em vista as diferentes dimensões de $\mathcal{R}_1$ e $\mathcal{R}_2$, a projeção

$$\begin{aligned} Proj_{(x,y)}(\mathcal{R}_2) \ = \ & \{(x, y) \in \mathbb{R}_+^{|E|+|V|} \ : \ (f, q, x, y) \in \mathcal{R}_2 \\ & \text{para algum } (f, q) \in \mathbb{R}_+^{\frac{|V|(|V|-1)(|A|-1)}{2}}\} \end{aligned}$$

é utilizada.

Proposição 2.5.1 *$\mathcal{R}_1$ e $Proj_{(x,y)}(\mathcal{R}_2)$ correspondem à mesma região poliedral.*

Prova 2.5.1 *Dado que o par de desigualdades (2.2)-(2.3) e (2.14)-(2.11) são idênticas, não precisaremos levá-las em consideração diretamente em nossa demonstração.*

$(\Rightarrow) Proj_{(x,y)}(\mathcal{R}_2) \subseteq \mathcal{R}_1$: É preciso mostrar que $(\bar{x}, \bar{y}) \in \mathcal{R}_1$ para qualquer $(\bar{x}, \bar{q}, \bar{f}, \bar{y}) \in \mathcal{R}_2$. Assim sendo, para chegar a uma contradição, seja $(\bar{x}, \bar{q}, \bar{f}, \bar{y}) \in \mathcal{R}_2$ e assuma que um par de vértices $i, j \in V$ existe tal que $i \in S \subset V$, $j \in V \setminus S$ e $\sum_{e \in \delta(S)} \bar{x}_e < 2(\bar{y}_i + \bar{y}_j - 1)$ se aplicam. Decorre então disso que a capacidade do corte mínimo separando i e j é maior que $2(\bar{y}_i + \bar{y}_j - 1)$ e, de acordo com o Teorema do Corte-Mínimo Fluxo-Máximo de FORD e FULKERSON (1956), o fluxo máximo de i à j não é maior que $2(\bar{y}_i + \bar{y}_j - 1) \leq 2\bar{q}_{ij}$. No entanto, desigualdades (2.9) seriam então violadas para $s = i$ e $t = j$, contradizendo nossa premissa de que $(\bar{x}, \bar{q}, \bar{f}, \bar{y}) \in \mathcal{R}_2$. Logo, $(\bar{x}, \bar{y})$ satisfaz as desigualdades (2.4) e, consequentemente, $(\bar{x}, \bar{y}) \in \mathcal{R}_1$.

$(\Leftarrow) \mathcal{R}_1 \subseteq Proj_{(x,y)}(\mathcal{R}_2)$: Para qualquer ponto $(\bar{x}, \bar{y}) \in \mathcal{R}_1$, devemos mostrar que existem $\bar{f}$ e $\bar{q}$ tais que $(\bar{x}, \bar{q}, \bar{f}, \bar{y}) \in \mathcal{R}_2$. Se tomarmos $\bar{q}_{ij} = max\{\bar{y}_i + \bar{y}_j - 1, 0\}$, as desigualdades (2.12), (2.13) e (2.17) são satisfeitas. Dessa forma, se existir $\bar{f}$ satisfazendo as desigualdades (2.9) e (2.18) para os valores selecionados de $\bar{q}$, obtemos o resultado. Para mostrar que este é o caso, note, de (2.4), que a capacidade do corte mínimo separando qualquer par distinto de vértices $i, j \in V$ é $2(\bar{y}_i + \bar{y}_j - 1)$. Portanto, pelo Teorema do Corte-Mínimo Fluxo-Máximo, o fluxo máximo com origem em i e destino em j deve ser igual à $2(\bar{y}_i + \bar{y}_j - 1) = 2\bar{q}_{ij}$. Com isso, para os valores selecionados de $\mathbf{q}$ e para todo par de vértices distintos $s, t \in V$, existe $\bar{f}_{ij}^{st}$ satisfazendo (2.9) e (2.18), e a demostração é então concluída. □

Vamos agora comparar $\mathcal{R}_1$ e $\mathcal{R}_3$ e, para isso, uma projeção

$$Proj_{(t,y)}(\mathcal{R}_3) = \{(t, y) \in \mathbb{R}_+^{|E|+|V|} \ : \ (h, y) \in \mathcal{R}_3,\ t_e = h_{(i,j)} + h_{(j,i)},\ e = \{i, j\} \in E\}$$

de $\mathcal{R}_3$ no espaço de variáveis de $\mathcal{R}_1$ é considerada.

Proposição 2.5.2 *$\mathcal{R}_1$ e $Proj_{(t,y)}(\mathcal{R}_3)$ correspondem à mesma região poliedral.*

Prova 2.5.2 *Dado que o par de desigualdades (2.2)-(2.3) e (2.22)-(2.23) são idênticas, não iremos levá-los em consideração diretamente em nossa demostração.*

$(\Rightarrow) Proj_{(t,y)}(\mathcal{R}_3) \subseteq \mathcal{R}_1$: Dado um ponto $(\bar{h}, \bar{y}, \bar{z}) \in \mathcal{R}_3$, a seguinte relação se aplica a todo $i \in S \subset V$ e $j \in V \setminus S$: $\sum_{a \in \delta^-(S)} \bar{h}_a \geq \bar{y}_i + \bar{y}_j - 1$

e $\sum_{a\in\delta^-(V\setminus S)} \bar{h}_a \geq \bar{y}_i + \bar{y}_j - 1$. *Somando as duas desigualdades, obtemos* $\sum_{a\in\delta^-(S)} \bar{h}_a + \sum_{a\in\delta^-(V\setminus S)} \bar{h}_a \geq 2(\bar{y}_i + \bar{y}_j - 1)$ *e, após algumas poucas manipulações algébricas, temos que* $\sum_{a\in\delta^-(S)} \bar{h}_a + \sum_{a\in\delta^+(S)} \bar{h}_a \geq 2(\bar{y}_i + \bar{y}_j - 1) \Rightarrow \sum_{e=\{k,l\}\in\delta(S)} (\bar{h}_{kl} + \bar{h}_{lk}) \geq 2(\bar{y}_i + \bar{y}_j - 1)$. *Por fim, substituindo por* $\bar{t}$, *obtemos* $\sum_{e\in\delta(S)} \bar{t}_e \geq 2(\bar{y}_i + \bar{y}_j - 1)$ *e portanto* $(\bar{t}, \bar{y}) \in \mathcal{R}_1$.

$(\Leftarrow)\mathcal{R}_1 \subseteq Proj_{(t,y)}(\mathcal{R}_3)$ **:** *Dado um ponto* $(\bar{x}, \bar{y}) \in \mathcal{R}_1$, *desigualdades* $\sum_{e\in\delta(S)} \bar{x}_e \geq 2(\bar{y}_i + \bar{y}_j - 1)$ *são válidas para qualquer* $i \in S \subset V$ *e* $j \in V \setminus S$. *Além disso, tomando* $\bar{h}_{kl} = \bar{h}_{lk} = \frac{\bar{x}_e}{2}$, *para todo* $e = \{k, l\} \in E$, *as seguintes relações se aplicam:*
(i) $\sum_{e\in\delta(S)} \bar{x}_e = \sum_{a\in\delta^-(S)} \bar{h}_a + \sum_{a\in\delta^+(S)} \bar{h}_a \geq 2(\bar{y}_i + \bar{y}_j - 1)$,
(ii) $\sum_{a\in\delta^-(S)} \bar{h}_a \geq \bar{y}_i + \bar{y}_j - 1$ *e*
(iii) $\sum_{a\in\delta^+(S)} \bar{h}_a \geq \bar{y}_i + \bar{y}_j - 1$.
Dessa forma, $(\bar{x}, \bar{y}) \in Proj_{(t,y)}(\mathcal{R}_3)$. □

Por fim, fechando essa seção, $\mathcal{R}_3$ e $\mathcal{R}_4$ são comparadas e, para tanto, uma projeção

$$Proj_{(h,y)}(\mathcal{R}_4) = \{(h, y) \in \mathbb{R}_+^{\frac{(|V|+1)|V|}{2}} : (h, y, z) \in \mathcal{R}_4 \text{ para algum } z \in \mathbb{R}_+^{|V|}\}$$

de $\mathcal{R}_4$ no espaço de variáveis de $\mathcal{R}_3$ é considerada. A demonstração é baseada em exemplos. Consideramos especificamente os dois grafos distintos $G = (V, E)$ descritos nas Figuras 2.7 e 2.8, acompanhados dos pontos de $\mathcal{R}_4$, para o primeiro grafo, e de $\mathcal{R}_3$, para o segundo.

Proposição 2.5.3 $\mathcal{R}_3$ *e* $Proj_{(h,y)}(\mathcal{R}_4)$ *não correspondem à mesma região poliedral.*

Prova 2.5.3 $(\Rightarrow)Proj_{(h,y)}(\mathcal{R}_4) \not\subset \mathcal{R}_3$ **:** *Considere o grafo* $G = (V, E)$ *apresentado na Figura 2.7 e o grafo suporte que o acompanha, o qual está associado a um ponto* $(\bar{h}, \bar{y}, \bar{z}) \in \mathcal{R}_4$. *As entradas não-nulas correspondentes a este ponto são:* $\bar{z}_1 = \bar{z}_{10} = 0.5$, $\bar{h}_{21} = \bar{h}_{13} = \bar{h}_{14} = \bar{h}_{35} = \bar{h}_{51} = \bar{h}_{57} = \bar{h}_{79} = \bar{h}_{9\,10} = \bar{h}_{10\,7} = \bar{h}_{10\,8} = \bar{h}_{86} = \bar{h}_{6\,10} = \bar{h}_{64} = \bar{h}_{42} = 0.5$, $\bar{y}_2 = \bar{y}_3 = \bar{y}_4 = \bar{y}_5 = \bar{y}_6 = \bar{y}_7 = \bar{y}_8 = \bar{y}_9 = 0.5$, *e* $\bar{y}_1 = \bar{y}_{10} = 1.0$. *Assim sendo, é fácil verificar que* $(\bar{h}, \bar{y}, \bar{z})$ *satisfaz todas as desigualdades em (2.38) e que portanto pertence à* $\mathcal{R}_4$. *Neste sentido, note que os arcos mais externos do grafo suporte correspondem à um circuito Hamiltoniano onde todo arco possui capacidade* 0.5. *Portanto, as desigualdades (2.28) e (2.29) são satisfeitas para qualquer subconjunto* $S \setminus V$ *contido em* $\{2, \ldots, 9\}$.

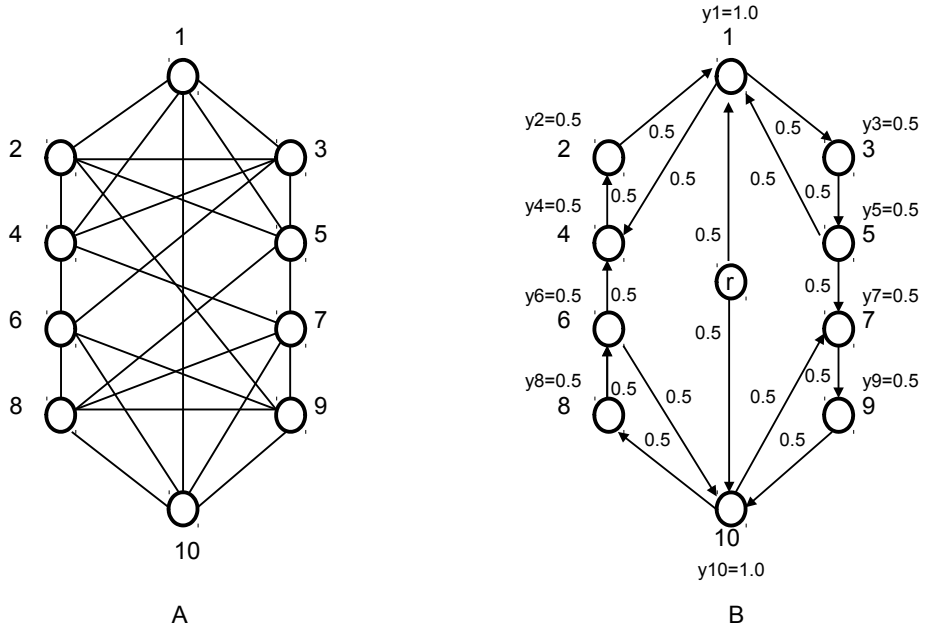

Figura 2.7: $G = (V, E)$ e uma solução correspondente de $\mathcal{R}_4$ e $\mathcal{R}_6$.

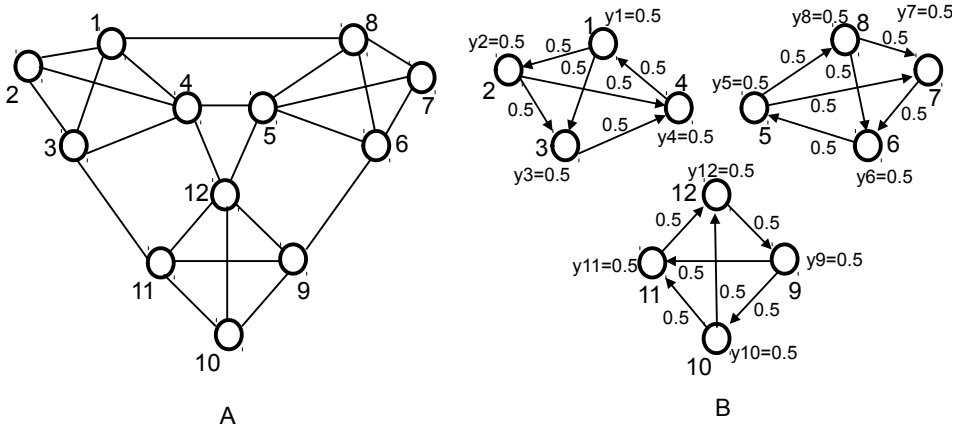

Figura 2.8: $G = (V, E)$ e uma solução correspondente de $\mathcal{R}_3$.

Da mesma forma, para subconjuntos $S \setminus V$ que contêm os vértices 1 e 10, $\bar{z}_1 = \bar{z}_{10} = 0.5$ garante que essas desigualdades também são satisfeitas. No entanto, aplicando $Proj_{(h,y)}(\mathcal{R}_4)$ à $(\bar{h}, \bar{y}, \bar{z})$, resulta em um ponto $(\bar{h}, \bar{y})$ que viola pelo menos uma das desigualdades de (2.26). Especificamente, viola a desigualdade (2.21), correspondente a $S = \{1, 2, 3, 4, 5\}$. Podemos então afirmar que $Proj_{(h,y)}(\mathcal{R}_4) \not\subset \mathcal{R}_3$.

$(\Leftarrow)\mathcal{R}_3 \not\subset Proj_{(x,y)}(\mathcal{R}_4)$: Considere o grafo $G = (V, E)$ apresentado na Figura 2.8 e o grafo suporte que o acompanha, o qual está associado a um ponto $(\bar{h}, \bar{y}) \in \mathcal{R}_3$. As entradas não-nulas correspondentes a este ponto são: $\bar{h}_{12} = \bar{h}_{13} = \bar{h}_{23} = \bar{h}_{24} = \bar{h}_{34} = \bar{h}_{41} = \bar{h}_{58} = \bar{h}_{57} = \bar{h}_{87} = \bar{h}_{86} = \bar{h}_{76} = \bar{h}_{65} = \bar{h}_{10\,11} = \bar{h}_{10\,12} = \bar{h}_{11\,12} = \bar{h}_{12\,9} = \bar{h}_{9\,11} = \bar{h}_{9\,10} = 0.5$ e $\bar{y}_1 = \bar{y}_2 = \bar{y}_3 = \bar{y}_4 = \bar{y}_5 = \bar{y}_6 = \bar{y}_7 = \bar{y}_8 = \bar{y}_9 = \bar{y}_{10} = \bar{y}_{11} = \bar{y}_{12} = 0.5$. Dessa maneira, verificar que $(\bar{h}, \bar{y})$ satisfaz as desigualdades (2.22),(2.23),(2.24), e (2.25) é imediato. Além disso, já que $y_i + y_j - 1 = 0$ para todo par de vertices $i, j \in V$ distintos, temos que $(\bar{h}, \bar{y})$ também satisfaz as desigualdades (2.21). Portanto, pode-se concluir que $(\bar{h}, \bar{y})$ pertence o $\mathcal{R}_3$. Nos restaria então mostrar que não existe valor para $\boldsymbol{z}$ tal que $(\bar{h}, \bar{y}, \bar{z}) \in \mathcal{R}_4$. Para isso, como ocorre um empate na escolha Γ_{min}, tomamos arbitrariamente $\Gamma_{min} = \Gamma_2$. Dessa forma, apenas as variáveis $\bar{z}_1$, $\bar{z}_2$, e $\bar{z}_3$, podem, a princípio, assumir um valor não-nulo. No entanto, não existem valores que permitam as desigualdades (2.28) e (2.29) serem satisfeitas para $S = \{1, 2, 3, 4\}$ e $i = 10$. Logo, $\mathcal{R}_3 \not\subset Proj_{(h,y)}(\mathcal{R}_4)$. $\square$

2.5.2 Caso 2-Vértice Conexo

Relações de dominância envolvendo as relaxações lineares das formulações para a variante 2-vértice conexa do PCD2CM serão discutidas a seguir.

Inicialmente iremos mostrar que a formulação (2.40) e a sua reformulação direcionadas, (2.45), não são equivalentes. Tendo em vista as diferentes dimensões de $\mathcal{R}_5$ e $\mathcal{R}_6$, a projeção

$$\begin{aligned} Proj_{(t,y)}(\mathcal{R}_6) \;=\; & \{(t, y) \in \mathbb{R}_+^{|E|+|V|} \;:\; (h, y, z) \in \mathcal{R}_5,\; t_e = h_{(i,j)} + h_{(j,i)}, \\ & e = \{i, j\} \in E, \;\text{ para algum } z \in \mathbb{R}_+^{|V|}\} \end{aligned}$$

de $\mathcal{R}_6$ no espaço de variáveis de $\mathcal{R}_5$ é considerada.

Consideramos especificamente os dois grafos distintos $G = (V, E)$ apresentados nas Figuras 2.7 e 2.9, acompanhados dos pontos de $\mathcal{R}_6$, para o primeiro grafo, e $\mathcal{R}_5$, para o segundo.

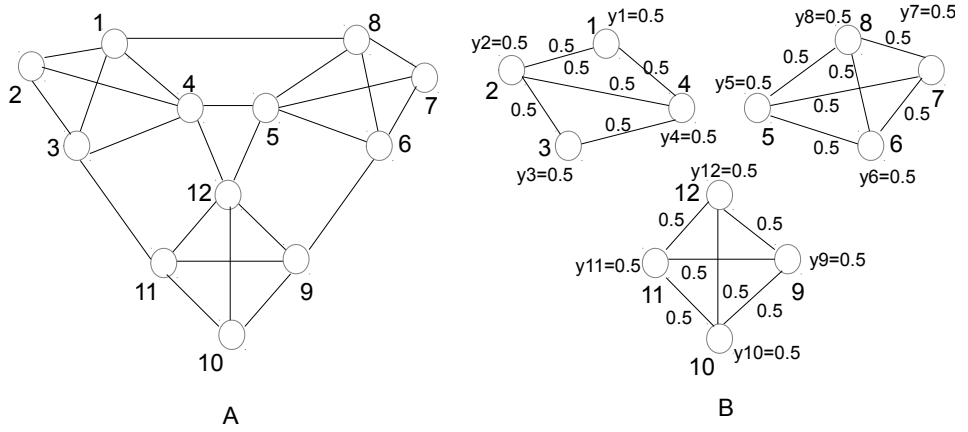

Figura 2.9: Um grafo e uma solução correspondente de $\mathcal{R}_5$ para o mesmo.

Proposição 2.5.4 *$\mathcal{R}_5$ e $Proj_{(t,y)}(\mathcal{R}_6)$ não correspondem à mesma região poliedral.*

Prova 2.5.4 $(\Rightarrow)Proj_{(t,y)}(\mathcal{R}_6) \not\subset \mathcal{R}_5$ *: Considere o grafo $G = (V, E)$ apresentado na Figura 2.7 e o grafo suporte que o acompanha, o qual está associado com um ponto $(\bar{h}, \bar{y}, \bar{z}) \in \mathcal{R}_6$. As entradas não-nulas correspondentes para este ponto são: $\bar{z}_1 = \bar{z}_{10} = 0.5$, $\bar{h}_{21} = \bar{h}_{13} = \bar{h}_{14} = \bar{h}_{35} = \bar{h}_{51} = \bar{h}_{57} = \bar{h}_{79} = \bar{h}_{9\,10} = \bar{h}_{10\,7} = \bar{h}_{10\,8} = \bar{h}_{86} = \bar{h}_{6\,10} = \bar{h}_{64} = \bar{h}_{42} = 0.5$, $\bar{y}_2 = \bar{y}_3 = \bar{y}_4 = \bar{y}_5 = \bar{y}_6 = \bar{y}_7 = \bar{y}_8 = \bar{y}_9 = 0.5$, e $\bar{y}_1 = \bar{y}_{10} = 1.0$. Como já visto na Proposição 2.5.3 a solução $(\bar{h}, \bar{y}, \bar{z})$ satisfaz todas as desigualdades contidas em $\mathcal{R}_4$. Assim sendo, é fácil verificar que $(\bar{h}, \bar{y}, \bar{z})$ também satisfaz as desigualdades em (2.44) e (2.42) e que portanto pertence à $\mathcal{R}_6$. Note que os arcos mais externos do grafo suporte correspondem à um circuito Hamiltoniano onde todo arco possui capacidade 0.5. Portanto, ao eliminarmos qualquer vértice desse grafo, existirá ao menos um caminho onde todo arco possui capacidade 0.5. Com isso, as desigualdades (2.44) são satisfeitas para qualquer subconjunto $S \setminus V$ contido em $\{2, \ldots 9\}$. Da mesma forma, para subconjuntos $S \setminus V$ que contêm os vértices 1 e 10, $\bar{z}_1 = \bar{z}_{10} = 0.5$ garante que essas desigualdades também são satisfeitas. No entanto, aplicando $Proj_{(t,y)}(\mathcal{R}_6)$ à $(\bar{h}, \bar{y}, \bar{z})$, resulta em um ponto $(\bar{t}, \bar{y})$ que viola pelo menos uma desigualdade de (2.4). Neste caso, a desigualdade (2.4) correspondendo à $S = \{1, 2, 3, 4, 5\}$. Podemos afirmar então que $Proj_{(t,y)}(\mathcal{R}_6) \not\subset \mathcal{R}_5$.*

$(\Leftarrow)\mathcal{R}_5 \not\subset Proj_{(t,y)}(\mathcal{R}_6)$ *: Considere o grafo $G = (V, E)$ apresentado na Figura 2.9 e o grafo suporte que o acompanha, o qual está associado a um ponto $(\bar{t}, \bar{y}) \in \mathcal{R}_5$.*

As entradas não-nulas correspondentes para este ponto são: $\bar{t}_{\{1,2\}} = \bar{t}_{\{1,3\}} = \bar{t}_{\{2,3\}} = \bar{t}_{\{2,4\}} = \bar{t}_{\{3,4\}} = \bar{t}_{\{4,1\}} = \bar{t}_{\{5,8\}} = \bar{t}_{\{5,7\}} = \bar{t}_{\{8,7\}} = \bar{t}_{\{8,6\}} = \bar{t}_{\{7,6\}} = \bar{t}_{\{6,5\}} = \bar{t}_{\{10,11\}} = \bar{t}_{\{10,12\}} = \bar{t}_{\{11,12\}} = \bar{t}_{\{12,9\}} = \bar{t}_{\{9,11\}} = \bar{t}_{\{9,10\}} = 0.5$ *e* $\bar{y}_1 = \bar{y}_2 = \bar{y}_3 = \bar{y}_4 = \bar{y}_5 = \bar{y}_6 = \bar{y}_7 = \bar{y}_8 = \bar{y}_9 = \bar{y}_{10} = \bar{y}_{11} = \bar{y}_{12} = 0.5$. *Dessa maneira, a verificação de que* $(\bar{t}, \bar{y})$ *satisfaz as desigualdades (2.2),(2.3),(2.5), e (2.6) é imediata. Além disso, já que* $y_i + y_j - 1 = 0$ *para todo par de vertices* $i, j \in V$ *distintos, temos que* $(\bar{t}, \bar{y})$ *também satisfaz as desigualdades (2.4) e (2.39). Portanto, pode-se concluir que* $(\bar{t}, \bar{y})$ *pertence à* $\mathcal{R}_5$. *Considerando* $\bar{h}_{kl} = \bar{h}_{lk} = \frac{\bar{t}_e}{2}$, *para todo* $e = \{k, l\} \in E$, *mostraremos que não existe valor para* $\boldsymbol{z}$ *tal que* $(\bar{h}, \bar{y}, \bar{z}) \in \mathcal{R}_6$. *Para isso, como há empate na escolha* Γ_{min}, *tomamos arbitrariamente* $\Gamma_{min} = \Gamma_2$. *Dessa forma, apenas as variáveis* $\bar{z}_1$, $\bar{z}_2$, *e* $\bar{z}_3$, *podem, a princípio, assumir um valor não-nulo. No entanto, não existem valores que permitam as desigualdades (2.28) e(2.29) serem satisfeitas para* $S = \{1, 2, 3, 4\}$ *e* $i = 10$. *Logo,* $\mathcal{R}_5 \not\subset Proj_{(t,y)}(\mathcal{R}_6)$. □

Por fim, serão comparadas a formulação (2.40) e a formulação multi-fluxos, (2.48) e, com isso, uma projeção

$$\begin{aligned} Proj_{(x,y)}(\mathcal{R}_7) \;\; = \;\; & \{(x,y) \in \mathbb{R}_+^{|E|+|V|} \; : \; (f,q,x,y) \in \mathcal{R}_7 \\ & \text{para algum } (f,q) \in \mathbb{R}_+^{\frac{|V|(|V|-1)(|A|-1)}{2}} \}. \end{aligned}$$

é considerada de $\mathcal{R}_7$ no espaço das variáveis de $\mathcal{R}_5$.

Proposição 2.5.5 $\mathcal{R}_5$ *e* $Proj_{(x,y)}(\mathcal{R}_7)$ *correspondem a mesma região poliedral.*

Prova 2.5.5 *Dado que, a região poliedral* $\mathcal{R}_5$ *é composta pela união da região* $\mathcal{R}_1$ *com as desigualdades 2.39 e a região poliedral* $\mathcal{R}_7$ *é composta pela união da região* $\mathcal{R}_2$ *com as desigualdades (2.47), na Proposição 2.5.1, já mostramos que* $\mathcal{R}_1$ *e* $Proj_{(x,y)}(\mathcal{R}_2)$ *correspondem a mesma região poliedral. Dessa forma, devemos levar em consideração somente as desigualdades (2.39), de* $\mathcal{R}_5$, *e (2.47), de* $\mathcal{R}_7$ *em nossa demonstração.*

$(\Rightarrow) Proj_{(x,y)}(\mathcal{R}_7) \subseteq \mathcal{R}_5$**:** *É preciso mostrar que* $(\bar{x}, \bar{y}) \in \mathcal{R}_5$ *para qualquer* $(\bar{x}, \bar{q}, \bar{f}, \bar{y}) \in \mathcal{R}_7$. *Partindo da Proposição 2.5.1, precisamos mostrar que* $(\bar{x}, \bar{y})$ *satisfaz também as desigualdades (2.39). Assim sendo, para chegar a uma contradição, seja* $(\bar{x}, \bar{q}, \bar{f}, \bar{y}) \in \mathcal{R}_7$ *e um vértice* $k \in V$ *qualquer, assumimos que um par de vértices* $i, j \in V \setminus \{k\}$ *existe tal que* $i \in S \subset V$, $j \in V \setminus S$, $\sum_{e \in \delta_{G_{\{V \setminus \{k\}\}}}(S)} \bar{x}_e < \bar{y}_i + \bar{y}_j - 1$ *e* $\sum_{e \in \delta(S)} \bar{x}_e \geq 2(\bar{y}_i + \bar{y}_j - 1)$ *sejam verdadeiras. Decorre então disso que a capacidade do corte mínimo separando* i *e* j *no grafo* $G_{\{V \setminus \{k\}\}}$ *é menor que* $\bar{y}_i + \bar{y}_j - 1$ *e, de acordo com o Teorema do Corte-Mínimo Fluxo-Máximo, o fluxo máximo de* i *à* j

em $G_{\{V\setminus\{k\}\}}$ não é maior que $\bar{y}_i + \bar{y}_j - 1 \leq \bar{q}_{ij}$. No entanto, pelas desigualdades (2.9), mais de uma unidade de fluxo teria que passar pelo vértice k, ou seja, $\sum_{i\in\Gamma_k,(ik)\in A} f_{ik}^{st} > 1$ para $s = i$ e $t = j$. Dessa forma, seriam então violadas as desigualdades (2.47) , contradizendo assim a nossa premissa que $(\bar{x}, \bar{q}, \bar{f}, \bar{y}) \in \mathcal{R}_7$. Logo, $(\bar{x}, \bar{y})$ satisfaz as desigualdades (2.39) e, consequentemente, $(\bar{x}, \bar{y}) \in \mathcal{R}_5$.

($\Leftarrow$)$\mathcal{R}_5 \subseteq Proj_{(x,y)}(\mathcal{R}_7)$: Para qualquer ponto $(\bar{x}, \bar{y}) \in \mathcal{R}_5$, devemos mostrar que existem $\bar{f}$ e $\bar{q}$ tais que $(\bar{x}, \bar{q}, \bar{f}, \bar{y}) \in \mathcal{R}_7$. Partindo da Proposição 2.5.1, precisamos mostrar que $\bar{f}$ satisfaz também as desigualdades (2.47). Tal como na Proposição 2.5.1, faremos $\bar{q}_{ij} = \bar{y}_i + \bar{y}_j - 1$. Dessa forma, se existir $\bar{f}$ satisfazendo as desigualdades (2.47) para aqueles valores selecionados de $\bar{q}$, obteremos assim o resultado. Para mostrar que este é o caso, note, de (2.4), que a capacidade do corte mínimo separando qualquer par distinto de vértices $i, j \in V$ é $2(\bar{y}_i + \bar{y}_j - 1)$. Além disso, pelas restrições (2.39), para uma escolha arbitrária de $k \in V$, o corte mínimo separando qualquer par distinto de vértices $i, j \in V \setminus \{k\}$ no grafo $G_{\{V\setminus\{k\}\}}$ é de pelo menos $\bar{y}_i + \bar{y}_j - 1$. Pelo Teorema do Corte-Mínimo Fluxo-Máximo, o fluxo máximo com origem em i e destino j deve ser igual a $2(\bar{y}_i + \bar{y}_j - 1) = 2\bar{q}_{ij}$ no grafo G e igual a $\bar{y}_i + \bar{y}_j - 1$ para o grafo $G_{\{V\setminus\{k\}\}}$. Como a única diferença entre os grafos G e $G_{\{V\setminus\{k\}\}}$ é a ausência das arestas que incidem em k e do vértice k, podemos afirmar que $\sum_{i\in\Gamma_k,(ik)\in A} f_{ik}^{st} \leq \bar{y}_i + \bar{y}_j - 1 \leq 1$ em G. Portanto, para os valores selecionados de $\mathbf{q}$ e para todo par de vértices distintos $s, t \in V \setminus \{k\}$, existe $\bar{f}_{ij}^{st}$ satisfazendo (2.47), e a demostração é então concluída. $\square$

Capítulo 3

Heurísticas Primais

Neste capítulo, serão apresentados dois métodos heurísticos desenvolvidos neste trabalho para a obtenção de soluções primal viáveis para ambas as variantes do PCD2CM. Em seguida, será apresentado um procedimento para tentar reduzir a cardinalidade das soluções viáveis encontradas.

3.1 Verificação de 2-Conexidade

Antes de apresentarmos as heurística desenvolvidas para as duas variantes do PCD2CM, faz-se necessário apresentar um procedimento responsável por verificar se um dado grafo $G = (V, E)$ é 2-aresta conexo, ou 2-vértice conexo, sendo esse essencial para a construção de soluções primal viáveis para o problema. Caso o grafo G não seja conexo, este procedimento fornece uma lista contendo todas as pontes, arestas de G que ao serem removidas desconectam o grafo, e todos as articulações, vértices de G que ao serem removidos desconectam o grafo.

Diversos algoritmos foram desenvolvidos com o propósito de verificar a 2-{aresta, vértice} conexidade de um grafo. Dentre eles destacam-se os algoritmos *Open Ear Decomposition* (LOVÁSZ, 1985), *Block-Cut Tree* (HARARY e PRINS, 1966), *Bipolar Orientation* (BRANDES, 2002), *s-t-Numbering* (BRANDES, 2002) e *Ear Decomposition* (LOVÁSZ, 1985). A maior parte desses algoritmos utiliza uma árvore proveniente de um algoritmo de busca em profundidade para obter valores chamados de *low-points*, introduzidos por TARJAN (1972). No entanto, neste trabalho, foi escolhido um procedimento introduzido por SCHMIDT (2013), que se baseia em caminhos ao invés de valores de *low-point*. A motivação dessa escolha vem da simplicidade e da facilidade de implementação desse procedimento.

O procedimento de Schmidt consiste em decompor um grafo conexo $G = (V, E)$ em um conjunto de caminhos e ciclos, denominados "cadeias". A partir dessa decomposição em cadeias, algumas propriedades irão determinar se G é ou não um grafo 2-aresta conexo, ou 2-vértice conexo.

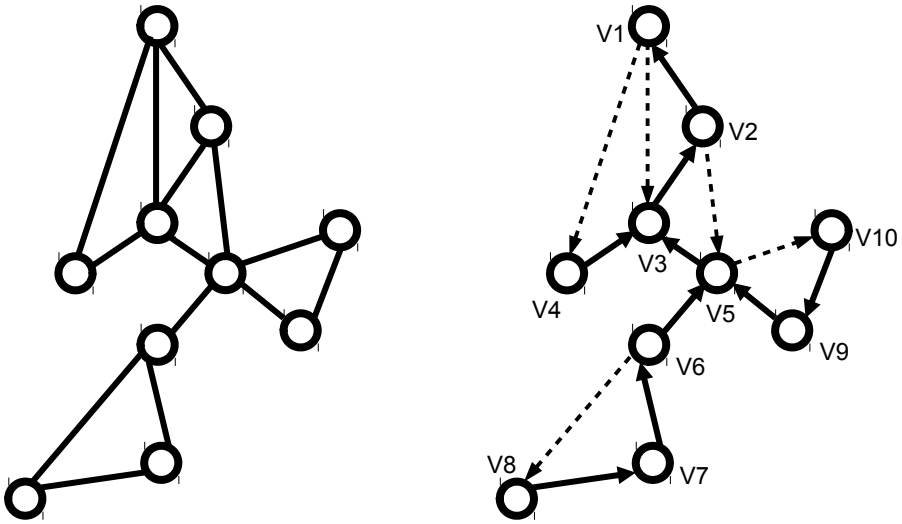

Figura 3.1: Grafo $G = (V, E)$, à esquerda, e árvore correspondente à busca em profundidade com suas *back-edges* em linhas tracejadas, à direita.

Inicialmente, neste procedimento, uma busca em profundidade é realizada, verificando dessa forma se o grafo é conexo. Caso não seja, o procedimento informa que G não é 2-aresta conexo e nem 2-vértice conexo. Alternativamente, se o grafo é conexo, uma árvore T com raiz em um vértice $r \in V$ é obtida através dessa busca em profundidade. Neste processo, a ordem em que os vértices são visitados é armazenada, de tal forma que o vértice u, por exemplo, é visitado na posição $IBP(u)$. Em seguida, o grafo G é orientado de tal forma que todos as arestas de G que também pertencem à árvore T, corresponderão, na orientação, à um arco direcionado para o vértice raiz r. Adicionalmente, todas as arestas que não pertencem a T são chamadas de *back-edges* e corresponderão a arcos orientados no sentido oposto de r. A Figura 3.1 ilustra o processo de orientação. Nessa figura, $r = v_1$ e os *back-edges* são indicados por linhas tracejadas. Pode-se observar que cada *back-edge* $e \in E$ faz parte de um ciclo orientado $C(e)$. Após a orientação de G, todos os seus vértices são inicializados como não-visitados.

A decomposição de G em cadeias é realizada seguindo a ordem crescente de IBP. Para cada *back-edge* e com extremidade v, o ciclo $C(e)$ é percorrido rotulando cada vértice percorrido como visitado, até que um vértice já rotulado como visitado for encontrado. Note que um percurso será um caminho ou um ciclo direcionado, denominado cadeia. Cada cadeia está associada a vértices e arestas, de G, e a i-ésima cadeia encontrada é identificada por C_i. Um total de $|E| - |V| + 1$ cadeias

serão encontradas nessa decomposição, uma vez que cada *back-edge* dará origem a exatamente uma cadeia. Note que a primeira cadeia C_1, se existir, será sempre um ciclo, já que, inicialmente, todos os vértices estão rotulados com não-visitados. Uma decomposição em cadeias é um conjunto $C = \{C_1, C_2, ..., C_{(|E|-|V|+1)}\}$. O Algoritmo (1) é um pseudo-código da decomposição em cadeia.

Dados: Um grafo simples $G = (V, E)$
Resultado: Uma decomposição em cadeias C para o grafo G ($C = \emptyset$ se G não for conexo).
$C = \emptyset$
Um Busca em Profundidade é realizada em G.
se *G é conexo* **então**
  $T = (V, E_t) \leftarrow$ Árvore definida pela busca em profundidade em G.
  IBP $\leftarrow$ Índices para os vértices de V com a ordem em que os vértices foram visitados.
  $BackEdges = \{\{u, v\} : \{u, v\} \in E$ e $\{u, v\} \notin E_t\}$
  $A = \{(u, v) : \{u, v\} \in E_t$ e $IBP(v) < IBP(u)\} \cup \{(u, v) : \{u, v\} \in BackEdges$ e $IBP(u) < IBP(v)\}$
  $D = (V, A)$ orientação de G definida por T e $BackEdges$.
  $i = 0$
  Rotular todos os vértices de V como não-visitados.
  para cada $(u, v) \in A$ *tal que* $\{u, v\} \in BackEdges$ **faça**
    Rotular u como visitado.
    $C_i \leftarrow$ Caminho iniciando em u e terminando no primeiro vértice rotulado como visitado.
    Rotular todos os vértices e arestas no caminho C_i como visitados.
    $i \leftarrow i + 1$
  fim
  $C = \{C_1, C_2, ..., C_{(|E|-|V|+1)}\}$
fim
Retorna C.

Algoritmo 1: Decomposição em cadeias de um grafo $G = (V, E)$.

Na referência SCHMIDT (2013), Schmidt introduz uma série de teoremas para caracterizar um grafo conexo $G = (V, E)$ como 2-aresta conexo ou 2-vértice conexo. Estes teoremas são discutidos a seguir.

Teorema 3.1.1 (SCHMIDT (2013)) *Seja C uma decomposição em cadeias de um grafo simples conexo $G = (V, E)$. O grafo G é 2-aresta conexo se e somente se o conjunto de arestas contidas em C particiona o conjunto de arestas E.*

Durante a decomposição em cadeias do grafo G, se além dos vértices de V, forem também rotuladas como visitadas todas as arestas de G visitadas, basta verificar, segundo o Teorema 3.1.1, se alguma aresta de G não foi rotulada para determinar se o grafo é, ou não, 2-aresta conexo. Com isso, um algoritmo pode ser definido para

determinar a 2-aresta conexidade de um grafo qualquer. Este algoritmo é expresso pelo pseudo-código Algoritmo 2.

Dados: Um grafo simples $G = (V, E)$
Resultado: Conjunto $Pontes \subset E$ das pontes do grafo G ($Pontes = \emptyset$ se G for 2-aresta conexo).
$C \leftarrow$ Decomposição em cadeias do grafo G.
$Pontes = \emptyset$
Rotular todas os arestas de E como não-visitados.
se $C \neq \emptyset$ **então**
  para cada $C_i \in C$ **faça**
    para cada $e \in C_i$ **faça**
      Rotular a aresta e como visitada .
    fim
  fim
  para cada $e \in E$ **faça**
    se *e está rotulado como não-visitado* **então**
      $Pontes \leftarrow Pontes \cup \{e\}$
    fim
  fim
fim
Retorna $Pontes$ (Se o grafo G for 2-aresta conexo, $Pontes = \emptyset$.)

Algoritmo 2: Verificação de 2-aresta conexidade de um grafo $G = (V, E)$.

Teorema 3.1.2 (SCHMIDT (2013)) *Seja C uma decomposição em cadeias de um grafo simples 2-aresta conexo $G = (V, E)$. O grafo G é 2-vértice conexo se e somente se C_1 é o único ciclo contido em C.*

Quando G é 2-aresta conexo, segundo o Teorema (3.1.2), se para alguma cadeia de sua decomposição em cadeias, além de C_1, tiver o primeiro e o último vértices iguais, o grafo G não é 2-vértice conexo.

Um algoritmo para determinar se um grafo G é 2-vértice conexo é expresso pelo pseudo-código Algoritmo 3.

3.2 Primeira Heurística

Dado um grafo conexo $G = (V, E)$, uma heurística simples e eficiente para obter conjuntos dominantes conexos de G foi proposta em LUCENA *et al.* (2010). Por outro lado, se o subgrafo de G induzido por um conjunto dominante conexo não é 2-{aresta,vértice} conexo, é possível expandi-lo, introduzindo vértices adicionais, na tentativa de obter um conjunto dominante com aquele grau de conexidade. Dessa forma, um algoritmo para gerar soluções primal viáveis para ambas as variantes do PCD2CM pode ser obtido através de dois procedimentos. O primeiro é uma

Dados: Um grafo simples $G = (V, E)$
Resultado: Um conjunto $Artc \subset V$ com as articulações de G ($Artc = \emptyset$ se G for 2-vértice conexo).
$C \leftarrow$ Decomposição em cadeias do grafo G.
$Pontes \leftarrow$ Verifica a 2-aresta conexidade do grafo G
$Artic = \emptyset$
se *G é 2-aresta conexo ($Pontes = \emptyset$)* **então**
  para cada $C_i \in C$ *e* $i > 1$ **faça**
    Seja s e $t \in C_i$ o primeiro e o último vértices da cadeia C_i, respectivamente.
    se $s = t$ **então**
      $Artic \leftarrow Artic \cup \{s\}$
    fim
  fim
senão
  para cada $e = \{u, v\} \in Pontes$ **faça**
    $Artic \leftarrow Artic \cup \{u, v\}$
  fim
fim
Retorna $Artc$ (Se o grafo G for 2-vértice conexo, $Artic = \emptyset$).

Algoritmo 3: Verificação de 2-vértice conexidade de um grafo G.

heurística simples e eficiente para a obtenção de um conjunto dominante conexo. Por sua vez, o segundo é um procedimento de expansão daquele conjunto, para obter, caso necessário, um conjunto dominante 2-{aresta,vértice} conexo.

Seja $G = (V, E)$ um grafo conexo e $i \in V$ um vértice escolhido arbitrariamente. O algoritmo heurístico para obtenção de um conjunto dominante conexo se inicia escolhendo um vértice $k \in \Gamma_i$, que dentre todos os vértices da vizinhança fechada Γ_i de i tenha o maior valor de $|\Gamma_k|$, ou seja, será escolhido o vértice vizinho de i com a maior quantidade de vizinhos em G. Escolhido o vértice k, o conjunto conexo inicial é dado por $W = \{k\}$ e o conjunto de vértices dominados por W é dado por $L = \Gamma_k \setminus \{k\}$. Caso W não seja um conjunto dominante, o algoritmo prossegue escolhendo um vértice $v \in L$ que tenha, na sua vizinhança, a maior quantidade de vértices não pertencentes à $W \cup L$. Escolhido v, o conjunto W é expandido com esse vértice. Da mesma forma, o conjunto L é atualizado inserindo-se a ele os vértices vizinhos de v que não pertencem ao conjunto W. O procedimento é finalizado assim que W for um conjunto dominante de G, ou seja, assim que $|W| + |L| = |V|$ resulta. Como o conjunto W inicial é unitário, é fácil verificar que o conjunto W final induz um grafo G_W conexo. Para tanto, basta observar que todo vértice adicionado a W é vizinho de algum vértice já inserido em W. Um pseudo-código para o procedimento descrito acima é apresentado no Algoritmo 4.

Passamos agora a descrever o segundo procedimento que compõe nosso primeiro algoritmo heurístico para encontrar soluções viáveis para o PCD2CM. Ele consiste

Dados: Um grafo simples $G = (V, E)$ e um vértice $u \in V$
Resultado: Um conjunto dominante conexo W.
$v = argmax\{|\Gamma_i| \ : \ i \in \Gamma_u\}$
$W = \{v\}$
$L = \Gamma_v \setminus \{v\}$
enquanto $|W| + |L| < |V|$ **faça**
  $v = argmax\{|\Gamma_j \setminus (W \cup L)| \ : \ j \in L\}$
  $W \leftarrow W \cup \{v\}$
  $L \leftarrow \Gamma_W \setminus W$
fim
Retorna W.

Algoritmo 4: Encontrar um conjunto dominante conexo W.

em expandir um conjunto dominante conexo W que não seja 2-{aresta, vértice} conexo, até que, eventualmente, ele assim se torne.

Neste procedimento, inicialmente, será utilizado o algoritmo descrito em SCHMIDT (2013), apresentado na Seção 3.1. Este, verifica se o grafo G_W é 2-{aresta,vértice} conexo. Caso não seja, pela estrutura do Algoritmo 2, as pontes do subgrafo G_W são especificadas pelo conjunto de arestas não-visitadas, *Pontes*. Dada uma ponte $\{u, v\} \in Pontes$, existe apenas um único caminho conectando u à v em G_W. No entanto, pelo Teorema 2.1.5, para que haja uma solução viável para o PCD2CM, é necessário que G seja 2-aresta conexo. Dessa forma, existe pelo menos um caminho distinto, que não utiliza a aresta $\{u, v\}$, conectando u à v em G. Um tal caminho pode ser obtido utilizando o algoritmo de DIJKSTRA (1959). Este, identifica um caminho mínimo em um grafo auxiliar $G' = (V, E \setminus \{u, v\})$, com custos zero para as arestas $\{i, j\} \in E \setminus \{u, v\}$ se $i, j \in W$, igual a um se $i \in W$ e $j \notin W$ e igual à $|E|$ se $i, j \notin W$. Com isso, o caminho encontrado prioriza vértices que pertencem ao conjunto dominante. Encontrado um caminho que conecte u à v sem passar por $\{u, v\}$, seus vértices não pertencentes a W são então inseridos neste conjunto. Após essa inserção, a aresta $\{u, v\}$ deixará de ser uma ponte para o novo grafo G_W.

Ao implementar o passo descrito acima, é possível que algumas das outras arestas pertencentes ao conjunto *Pontes* tenham deixado de serem pontes. Portanto, o procedimento de verificação de 2-aresta conexidade deve ser executado novamente e um novo conjunto *Pontes* deve ser obtido. Este procedimento é repetido até que G_W torne-se 2-aresta conexo.

O Algoritmo 5 descreve um pseudo-código para o procedimento heurístico descrito acima.

Se quisermos obter um grafo G_W que seja 2-vértice conexo, é necessário eventualmente inserir vértices em W até que G_W não contenha articulações, ou seja, vértices que ao serem removidos desconectam o grafo. O Algoritmo 3, ao ser apli-

Dados: Um grafo simples $G = (V, E)$
Resultado: Um conjunto dominante 2-aresta conexo W.
$W^* = V$
para cada $i \in V$ **faça**
  Encontrar o conjunto conexo W iniciado com o vértice i.
  Verifica a 2-aresta conexidade de G_W e retorna o conjunto $Pontes$.
  enquanto $Pontes \neq \emptyset$ **faça**
    $\{u, v\} \in Pontes$
    $G' = (V, E \setminus \{u, v\})$
    $C \leftarrow$ Conjunto dos vértices do caminho de custo mínimo conectando u à v em G'.
    $W \leftarrow W \cup C$
    Verifica a 2-aresta conexidade de G_W e retorna o conjunto $Pontes$.
  fim
  se $|W^*| > |W|$ **então**
    $W^* \leftarrow W$
  fim
fim
Retorna W^*.

Algoritmo 5: Primeira heurística primal para o PCD2CM 2-aresta conexo.

cado para verificar a 2-vértice conectividade do grafo G_W, retorna um conjunto $Artic \subset V$. Este conjunto contém todos os vértices de W que são articulações de G_W, caso este não seja 2-vértice conexo. Dado $v \in Artic$, são escolhidas duas arestas, $\{u_1, v\}, \{u_2, v\} \in E(W)$, adjacentes a v em G_W. Para que v deixe de ser uma articulação de G_W, deverão ser adicionados a W os vértices de um caminho em G que conecte u_1 a u_2 e envolva a menor quantidade de vértices em $V \setminus W$ e ao mesmo tempo não utilize as arestas $\{u_1, v\}$ e $\{u_2, v\}$ e o vértice v. Esse caminho pode ser obtido utilizando o algoritmo de DIJKSTRA (1959), para encontra o menor caminho conectando u_1 a u_2 no grafo auxiliar $G'' = (V \setminus \{v\}, E \setminus \{\{u_1, v\}, \{u_2, v\}\})$, com custos de arestas similares àqueles do grafo auxiliar G'. O Algoritmo 6 descreve um pseudo-código para o procedimento heurístico descrito acima.

A construção de um conjunto dominante conexo se inicia com a escolha arbitrária de um vértice do grafo G. Dessa forma, dependendo da escolha feita, o conjunto dominante 2-{aresta,vértice} conexo encontrado pode ter maior ou menor cardinalidade. Dessa forma, é interessante executar o procedimento, tomando alternadamente cada vértice de V como vértice inicial. Assim procedendo, $|V|$ soluções primal viáveis são obtidas para o PCD2CM e escolhemos a melhor delas.

Dados: Um grafo simples $G = (V, E)$
Resultado: Um conjunto dominante 2-vértice conexo W.
$W^* = V$
para cada $i \in V$ **faça**
  Encontrar o conjunto conexo W iniciado com o vértice i.
  Verifica a 2-vértice conexidade de G_W e retorna o conjunto $Artic$.
  enquanto $Artic \neq \emptyset$ **faça**
    $v \in Artic$ tal que existam $u_1, u_2 \in W$ e $\{u_1, v\}, \{u_2, v\} \in E$
    $G'' = (V \setminus \{v\}, E \setminus \delta(v))$
    $C \leftarrow$ Conjunto dos vértices do caminho de custo mínimo conectando u_1 à u_2 em G''.
    $W \leftarrow W \cup C$
    Verifica a 2-vértice conexidade de G_W e retorna o conjunto $Artic$.
  fim
  se $|W^*| > |W|$ **então**
    $W^* \leftarrow W$
  fim
fim
Retorna W^*.

Algoritmo 6: Primeira heurística primal para o PCD2CM 2-vértice conexo.

3.3 Segunda Heurística

O Teorema 2.1.4 fornece uma ideia que é central para o desenvolvimento de um procedimento para construção de subgrafos 2-vértice conexos.

Com base nesse teorema, dado um grafo 2-vértice conexo $G = (V, E)$ e um subconjunto $W^0 \subset V$ tal que G_{W^0} é 2-vértice conexo, podemos reconstruir G a partir de G_{W^0} inserindo sucessivos caminhos de G em G_{W^0}. Genericamente, à cada iteração $i \in \{0, 1, ..., n\}$, inserimos caminhos $P^i = \{p_1^i, p_2^i, ..., p_m^i\} \subset V$ em G_{W^i}, tais que $P^i \cap W^i = \{p_1^i, p_m^i\}$, ou seja, tomamos $G_{W^{i+1}} = (W^{i+1} = W^i \cup P^i, E(W^{i+1}))$, até que $W^n = V$.

Usando a ideia acima, desenvolvemos uma segunda heurística para encontrar soluções primal viáveis para o PCD2CM. Esta consiste em expandir um subconjunto $W \subset V$ inicial, cujo subgrafo G_W é 2-vértice conexo, até que este se torne um conjunto dominante 2-vértice conexo de G.

Para encontrar um subgrafo 2-vértice conexo inicial, a heurística é inicializada por um vértice $i \in V$ arbitrariamente escolhido. Feito isso, encontramos o caminho com o menor número de vértices ligando i a $j \in (\Gamma_i \setminus \{i\})$, sem utilizar a aresta $\{i, j\}$. Este caminho pode ser obtido aplicando o algoritmo de DIJKSTRA (1959), sobre um grafo auxiliar $G'' = (V, E \setminus \{i, j\})$, com custo igual a um em todas as arestas. Dentre todos os caminhos encontrados, para os diferentes valores de j, selecionamos aquele de menor cardinalidade e introduzimos os seus vértices ao conjunto inicial W. Sendo assim, G_W é o menor subgrafo 2-vértice conexo contendo o vértice i. O

conjunto dos vértices dominados por W, L, é definido como $L = \Gamma_W \setminus W$.

Em seguida, vértices adicionais são inseridos em W até que este se torne dominante 2-vértice conexo. Como nosso objetivo é encontrar W com menor cardinalidade possível, o vértice de L com o maior número de vizinhos em $V \setminus (W \cup L)$ é o próximo a ser escolhido para entrar em W. Dessa forma, um par de vértices $u, v \in V$ tal que $u \in W$, $v \in L$, $\{u, v\} \in E$ e $|\Gamma_v \setminus (W \cup L)|$ tenha o máximo valor, são selecionados. Para inserir v em W e ainda assim manter a 2-vértice conexidade de G_W, será usada a ideia contida no Teorema 2.1.4. Ou seja, o caminho selecionado para aumentar o conjunto dominante conectará o vértice u a um vértice $l \in \Gamma_u \cap W$ passando pelo vértice v, usando a menor quantidade de vértices de $V \setminus W$. Uma vez adicionados ao conjunto dominante, o conjunto L é atualizado como $L = \Gamma_W \setminus W$. O procedimento é repetido até que $|W| + |L| = |V|$ resulte.

Como a heurística inicialmente seleciona um vértice $i \in V$ arbitrário, o procedimento é repetido utilizando cada $i \in V$ como vértice inicial. Por fim, a solução que apresenta o menor valor de $|W|$ é selecionada. O Algoritmo 7 é um pseudo-código da heurística que acabamos de descrever.

Dados: Um grafo simples $G = (V, E)$
Resultado: Um conjunto dominante 2-vértice conexo W.
$L' = \{v \in V \ : \ |\delta(v)| = 1\}$
$W' = V \setminus L'$
para cada $i \in V$ **faça**
  $W \leftarrow$ Conjunto dos vértices pertencentes ao menor ciclo contendo i no grafo G.
  $L = \Gamma_W \setminus W$
  enquanto $|W| + |L| < |V|$ **faça**
    $\{u, v\} = \text{argmax}\{|\Gamma_v \setminus (W \cup L)| \ : \ v \in L, u \in W, \{u, v\} \in E\}$
    para cada $j \in (W \cap \Gamma_u) \setminus \{u\}$ **faça**
      $C_j \leftarrow$ Conjunto dos vértices do menor caminho ligando v à j.
    fim
    $C = \text{argmin}\{|C_j| \ : \ j \in (W \cap \Gamma_u) \setminus \{u\}\}$
    $W = W \cup C$
    $L = \Gamma_W \setminus W$
  fim
  se $|W'| > |W|$ **então**
    $W' \leftarrow W$
  fim
fim

Algoritmo 7: Segunda heurística primal para a variante 2-vértice conexa do PCD2CM.

Para que este procedimento possa ser utilizado para encontrar soluções viáveis

para a variante 2-aresta conexa do problema, algumas adaptações devem ser feitas. Especificamente, a cada iteração, inserimos em W os vértices do caminho de menor cardinalidade conectando u a v, sem entretanto utilizar a aresta $\{u, v\}$ e não havendo a necessidade de utilizar o vértice $l \in \Gamma_u \cap W$. O Algoritmo 8 é um pseudo-código para variante 2-aresta conexa da heurística.

Dados: Um grafo simples $G = (V, E)$
Resultado: Um conjunto dominante 2-aresta conexo W.
$L' = \{v \in V \ : \ |\delta(v)| = 1\}$
$W' = V \setminus L'$
para cada $i \in V$ **faça**
  $W \leftarrow$ Conjunto dos vértices pertencentes ao menor ciclo contendo i no grafo G.
  $L = \Gamma_W \setminus W$
  enquanto $|W| + |L| < |V|$ **faça**
    $\{u, v\} = \text{argmax}\{|\Gamma_v \setminus (W \cup L)| \ : \ v \in L, u \in W, \{u, v\} \in E\}$
    $C \leftarrow$ Conjunto dos vértices do menor caminho ligando u à v sem utilizar a aresta $\{u, v\}$.
    $W = W \cup C$
    $L = \Gamma_W \setminus W$
  fim
  se $|W'| > |W|$ **então**
    $W' \leftarrow W$
  fim
fim

Algoritmo 8: Segunda heurística primal para o PCD2CM 2-aresta conexo.

3.4 Redução da Cardinalidade de W.

É natural que, um conjunto W, obtido por um procedimento heurístico como os que acabamos de descrever, não seja minimal. Quando isso se aplica, podemos então tentar reduzir sua cardinalidade, mantendo entretanto as exigências de conexidade. Dessa maneira, dado $u \in W$, se existir $v \in L$ tal que $\Gamma_v \cap W = \{u\}$, o conjunto $W \setminus \{u\}$ não é dominante. Sendo assim, $NR = \{v \in L \ : \ |\Gamma_v \cap W| = 1\}$ é definido como o conjunto dos vértices de L que compartilham arestas com apenas um vértice de W. Com isso, qualquer vértice pertencente ao conjunto $R \subseteq W$ tal que $\Gamma_R \cap NR = \emptyset$ pode ser removido de W sem que este deixe de ser dominante. Note que só é possível remover um vértice de R, pois a remoção de múltiplos vértices não garante que o conjunto resultante seja dominante. Dessa forma, a atualização dos conjuntos NR e R após a remoção de cada vértice é necessária.

Baseado nessa ideia, foi desenvolvido um procedimento para tentar reduzir a cardinalidade da solução obtida pelo procedimento heurístico. Este procedimento consiste em, dado uma solução W, construir o conjunto R, tal como descrito anteriormente, e se existir um vértice $u \in R$, tal que $G_{\{W \setminus \{u\}\}}$ é 2-vértice(aresta) conexo, fazemos $W \leftarrow W \setminus \{u\}$. Repetimos esse processo até que $R = \emptyset$ ou se para todo $u \in R$, $G_{\{W \setminus \{u\}\}}$ não for 2-aresta(vértice) conexo. Apresentamos no Algoritmo 9 um pseudo-código para este procedimento.

Dados: Um grafo $G = (V, E)$ e um conjunto dominante 2-vértice (aresta) conexo $W \subseteq V$.
Resultado: Um conjunto dominante 2-vértice (aresta) conexo W.
$L = V \setminus W$
$NR = \{v \in L \ : \ |\Gamma_v \cap W| = 1\}$
$R = \{v \in W \ : \ \Gamma_v \cap NR = \emptyset\}$
enquanto $R \neq \emptyset$ **faça**
  $u \in R$
  se $G_{W \setminus \{u\}}$ *é 2-vértice(aresta) conexo* **então**
    $W \leftarrow W \setminus \{u\}$
    $NR = \{v \in L \ : \ |\Gamma_v \cap W| = 1\}$
    $R = \{v \in W \ : \ \Gamma_v \cap NR = \emptyset\}$
  senão
    $R \leftarrow R \setminus \{u\}$
  fim
fim

Algoritmo 9: Procedimento para redução de soluções viáveis para o PCD2CM.

3.5 Experimentos Computacionais Preliminares

Com o objetivo de decidir qual heurística primal será utilizada como parte integrante de nossos algoritmos *Branch and Cut*, efetuamos um pequeno teste computacional para identificar qual das abordagens gera limitantes superiores de melhor qualidade. Implementamos dois algoritmos HEU_1 e HEU_2, associados, respectivamente, à primeira e à segunda heurísticas construtivas descritas neste capítulo. Ambos os algoritmos contam com o procedimento para redução de cardinalidade. Para esse experimento, geramos pseudo-aleatoriamente um conjunto teste contendo grafos $G = (V, E)$ 2-vértice conexos com $|V| = \{30, 50, 70, 100, 120, 200\}$ e porcentagem de densidade do conjunto de arestas $\{5\%, 10\%, 25\%, 50\%, 70\%\}$. Os grafos foram inicializados com um ciclo Hamiltoniano $1, \ldots, |V|$. Então, arestas geradas pseudo-aleatoriamente foram introduzidas em G até que a densidade exigida pelo

grafo fosse atingida. Para os grafos assim gerados, a existência de um conjunto dominante 2-{aresta,vértice} conexo é garantida.

Os algoritmos foram implementados em linguagem C++ e foram executados em uma máquina com processador Intel Xeon 3.0 GHz e 4 GBits de memória RAM.

A Tabela 3.1 apresenta, para ambas variantes do problema, os valores das soluções primais obtidas para as instâncias do conjunto teste pelas heurísticas HEU_1 e HEU_2. Nesta tabela, as duas primeiras colunas informam, respectivamente, o número de vértices e densidade dos grafos associados às instâncias teste. A seguir, dois blocos de 4 colunas, cada, são utilizados para descrever os resultados obtidos para as variantes 2-aresta conexa e 2-vértice conexa das heurísticas. Cada um desses blocos é dividido em dois conjuntos de duas colunas. As duas primeiras estão associadas à heurística HEU_1 e as duas últimas à heurística HEU_2. Em cada caso, a cardinalidade do conjunto W obtido é indicada, assim como o tempo de CPU, em segundo, demandado para encontrá-las.

Como podemos observar na Tabela 3.1, a diferença entre as soluções primais produzidas pelos algoritmos não é muito grande. O tempo de CPU gasto por ambos também não apresenta uma diferença significativa. No entanto, o algoritmo HEU_2, correspondente a segunda heurística proposta neste trabalho, apresentou resultados um pouco melhores para algumas instâncias, tanto para a variante 2-aresta conexa, quanto para a variante 2-vértice conexa. A partir deste experimento utilizaremos a segunda heurística construtiva como parte dos algoritmos de solução exata descritos no próximo capítulo.

Evidentemente esse pequeno teste computacional não é suficiente para afirmar conclusivamente que uma heurística é melhor que a outra. Testes mais elaborados devem ser conduzidos para tal efeito. No entanto, aprofundar ainda mais o estudo sobre as heurísticas primais fugiria do escopo de nosso trabalho. Deixamos, então, essa tarefa para trabalhos futuros dedicados a esse propósito.

		2-Aresta Conexo				2-Vértice Conexo			
		HEU_1		HEU_2		HEU_1		HEU_2	
$\|V\|$	dens.(%)	$\|W\|$	$t(s)$	$\|W\|$	$t(s)$	$\|W\|$	$t(s)$	$\|W\|$	$t(s)$
30	10	20	0.02	20	0.02	19	0.02	19	0.02
	25	7	0.02	7	0.02	7	0.02	7	0.02
	50	3	0.04	3	0.03	3	0.05	3	0.04
	70	3	0.07	3	0.06	3	0.08	3	0.08
50	5	18	0.03	18	0.03	18	0.05	18	0.04
	10	10	0.05	10	0.05	10	0.14	10	0.13
	25	7	0.13	6	0.11	6	0.3	6	0.31
	50	3	0.33	3	0.31	3	0.31	3	0.32
	70	3	0.59	3	0.49	3	0.59	3	0.58
70	5	33	0.37	34	0.33	33	0.44	33	0.38
	10	15	0.21	16	0.19	16	0.25	16	0.22
	25	7	0.42	7	0.4	7	0.45	7	0.44
	50	4	1.14	4	1.2	4	1.25	4	1.23
	70	3	2.05	3	2.05	3	2.15	3	2.13
100	5	31	0.82	34	0.81	32	0.8	34	0.73
	10	16	0.7	17	0.75	17	0.98	16	1.64
	25	8	1.54	7	1.56	7	1.55	7	1.54
	50	4	4.9	4	4.8	4	4.9	4	4.9
	70	3	9.63	3	8.89	3	9.8	3	9.75
120	5	33	1.37	33	1.38	33	1.22	33	1.23
	10	17	1.2	16	1.32	17	1.24	16	1.24
	25	7	3.04	7	3.02	7	3.03	7	3.08
	50	4	10.71	4	10.55	4	9.98	4	10.03
	70	3	17.7	3	16.8	3	18.78	3	20.3
150	5	33	2.59	32	2.6	32	2.14	32	2.15
	10	18	2.47	17	2.5	17	2.46	17	2.45
	25	9	7.62	8	6.9	8	7.77	7	7.78
	50	5	31.3	5	21.88	5	24.5	5	26.95
	70	3	69.16	3	52.7	3	69	3	67.5
200	5	31	5.15	33	4.89	31	5.15	31	4.63
	10	19	20.53	19	21	20	27	20	26.81
	25	9	22.89	8	24.6	9	27.56	8	27.28
	50	5	132.72	5	93.75	5	139	5	153.3
	70	3	219.86	3	301	3	224	3	356

Tabela 3.1: Testes computacionais preliminares para HEU_1 e HEU_2.

Capítulo 4

Algoritmos de Solução Exata

Por volta de 1960, os trabalhos de CROES (1958) e LAND e DOIG (1960) deram origem a uma classe de algoritmos chamados de *Branch and Bound*. Tais algoritmos se destinam à solução exata de Problemas de Otimização Inteira ou Combinatória. O termo *Branch and Bound*, cunhado por LITTLE *et al.* (1963), é composto pelas palavras *Branch* (Ramificação) e *Bound* (Poda), e evidencia as duas estratégias básicas empregadas para a resolução do problema.

A primeira estratégia empregada, o *Branching*, consiste em particionar o espaço de solução do problema em subespaços disjuntos, onde o problema considerado em cada subespaço é teoricamente mais fácil de ser resolvido que o problema original. Já a segunda estratégia, o *Bounding*, calcula limites duais para cada um dos problemas considerados em cada subespaço. A combinação dessas duas estratégias permite elaborar um esquema de enumeração inteligente , também chamado de enumeração implícita, objetivando encontrar a solução do problema sem a necessidade de enumerar todas as soluções viáveis para o mesmo.

Dado o seguinte problema geral de otimização

$$z = \text{Min}\{cx \ : \ x \in S\}, \tag{4.1}$$

segundo a Proposição 4.0.1, a solução para um problema geral 4.1 pode ser obtida através da resolução de subproblemas menores e mais fáceis de resolver.

Proposição 4.0.1 *Seja $S = S_1 \cup ... \cup S_K$ uma decomposição do conjunto S em subconjuntos menores, e seja $z^k = Min\{cx \ : \ x \in S_k\}$ para $k = 1, ..., K$. Então $z = Min\{z^1, ..., z^K\}$.*

Informações sobre os valores obtidos para os limitantes dos subproblemas considerados, podem ser integradas e utilizados para obter melhores limitantes para o problema original como um todo. A proposição que se segue formaliza esta ideia.

Proposição 4.0.2 *Seja* $S = S_1 \cup ... \cup S_K$ *uma decomposição do conjunto* S *em subconjuntos menores, e seja* $z^k = Min\{cx : x \in S_k\}$ *para* $k = 1, ..., K$, $\underline{z}^k$ *é um limitante inferior para o valor de* z^k *e* $\bar{z}^k$ *é um limitante superior para o valor de* z^k. *Então* $\underline{z} = Min\{\underline{z}^1, ..., \underline{z}^K\}$ *é um limitante inferior para o valor de* z *e* $\bar{z} = Min\{\bar{z}^1, ..., \bar{z}^K\}$ *é um limitante superior para o valor de* z.

Sejam $\underline{z}_0$ e $\bar{z}_0$, respectivamente, os melhores limitantes inferior e superior que podem ser obtidos para o problema (4.1). Caso $\underline{z}_0 < \bar{z}_0$, uma enumeração implícita é iniciada. Dessa forma, utilizando um critério qualquer de particionamento, S é dividido em K conjuntos mutuamente disjuntos S_1,S_2,...,S_K. Associados a estes, temos K novos problemas, $z^k = \text{Min}\{cx : x \in S_k\}$ a resolver. A solução de menor custo assim obtida define uma solução ótima para nosso problema original, (4.1). Idealmente os problemas resultantes de uma partição devem ser mais fáceis de resolver que o problema original. Caso isso não se verifique, a enumeração implícita prossegue particionando cada um dos K subconjuntos em subconjuntos ainda menores, e assim sucessivamente. As sucessivas partições podem ser representadas na forma de uma árvore, tal como aquela apresentada na Figura (4.1), chamada de árvore de enumeração. Cada um dos subproblemas está associado a um nó dessa árvore, assim como os seus limitantes inferiores e superiores correspondentes.

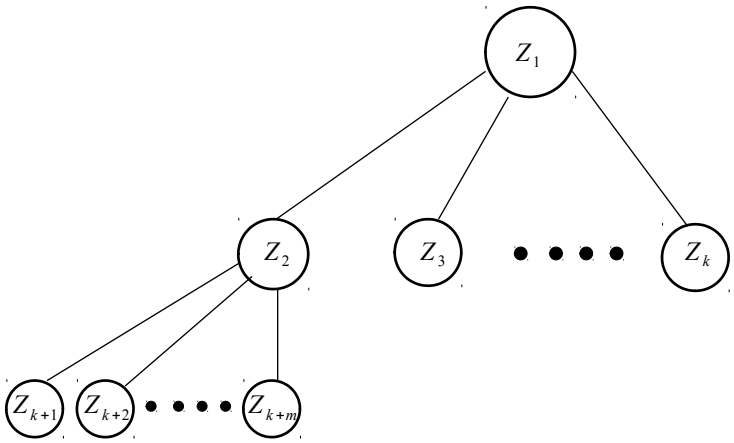

Figura 4.1: Árvore de enumeração.

Seja $\underline{z}_i$ um limitante inferior obtido para o problema definido sobre uma região S_i, se $\underline{z}_i \geq \bar{z}$, não é necessário particionar a região S_i, uma vez que qualquer solução para um problema definido a partir de um subconjunto de S_i terá o valor da função objetivo maior ou igual à $\underline{z}_i$. Com isso a solução do problema não será encontrada na região S_i, excluída assim do espaço de soluções (poda). Por outro lado, suponha que para um dado nó j, cujo problema é definido sobre o conjunto S_j, foram obtidos

limitantes $\underline{z}_j < \bar{z}_j \leq \bar{z}$. Neste caso, ainda há a possibilidade da solução ótima para o problema (4.1) pertencer ao conjunto S_j. Por conseguinte, a região S_j deve ser particionada (ramificada) em subconjuntos. Estes, por sua vez, terão limitantes associados tão ou mais apertados que aqueles correspondentes ao nó j. Além disso, o limitante superior $\bar{z}$ para o problema original poderá ser atualizado para $\bar{z}_j$.

O procedimento de *Branch and Bound* (Ramificação e Poda) continua até que, para todos os nós da árvore de enumeração, um limitante inferior obtido naquele nó seja maior ou igual ao melhor limite superior obtido para o problema, ou que não seja possível obter uma solução viável nesse nó, chamado de poda por inviabilidade.

O método de *Branch and Bound* foi descrito, até agora, da forma mais geral possível, não tendo sido definido um critério específico de ramificação e nem a forma de obter limitantes. A generalidade desse método permite a elaboração de diversos algoritmos, de solução, específicos para a resolução de um dado problema, ou de uma classe de problemas. Suponha, por exemplo, um Problema de Programação Inteira

$$z = \text{ Min}\{cx \ : \ Ax \leq b, x \in \mathbb{Z}^n\}, \tag{4.2}$$

onde $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{R}^n$. A região $S = \{x \in \mathbb{Z}^n \ : \ Ax \leq b\}$ pode ser particionada em dois subconjuntos disjuntos

$$S_1 = S \cap \{x_i \leq l_i\} \quad \text{e} \quad S_2 = S \cap \{x_i \geq l_i + 1\}, \tag{4.3}$$

para $l_i \in \mathbb{Z}$. Definindo, assim, os problemas

$$z_1 = \text{ Min}\{cx \ : \ Ax \leq b, x_i \leq l_i, x \in \mathbb{Z}^n\} \tag{4.4}$$

e

$$z_2 = \text{ Min}\{cx \ : \ Ax \leq b, x_i \geq l_i + 1, x \in \mathbb{Z}^n\} \tag{4.5}$$

associados respectivamente aos nós 1 e 2 da árvore de enumeração. Nesse caso, como a ramificação dá origem a dois nós, esta recebe o nome de árvore de enumeração binária, e o esquema de ramificação se denomina ramificação por variáveis. Um limitante superior pode ser obtido através de uma heurística desenvolvida para o problema, ou pode estar associado ao valor da função objetivo da primeira solução viável eventualmente encontrada ao longo da enumeração. Já o limitante inferior para cada nó pode ser obtido resolvendo-se a relaxação linear do problema considerado que, como visto anteriormente nesse texto, define um limitante inferior para o Problema de Programação Inteira.

Uma outra classe de algoritmos é obtida quando o Método de Plano de Cortes é inserido no Método de *Branch and Bound*. Como resultado, limitantes duais de

melhor qualidade são obtidos em cada nó da árvore de enumeração e o processo de enumeração é acelerado ao se podar mais nós da árvore de busca. Esta classe de algoritmos é chamada de *Branch and Cut* (GRÖTSCHEL *et al.*, 1984; PADBERG e RINALDI, 1991).

Nos algoritmos *Branch and Cut*, em cada nó i da árvore de enumeração, os limitantes duais $\underline{z}_i$ são obtidos através da resolução da relaxação linear associada à região S_i definida para esse nó. Caso a solução ótima assim obtida seja fracionária, desigualdades válidas $\mathcal{D}$ violadas pela solução são identificadas e adicionadas à região S_i, tomando-se então $S_i \leftarrow S_i \cap \mathcal{D}$. Dessa forma, um novo limite inferior, teoricamente melhor que o anterior, será obtido através da resolução da relaxação linear para essa nova região, mais apertada.

As desigualdades válidas aludidas acima são identificadas por meio da resolução de um problema de separação. Todo algoritmo capaz de encontrar uma desigualdade válida violada pela solução ótima da relaxação linear corrente, caso exista uma, é chamado de algoritmo exato de separação. Se não houver garantia que toda desigualdade válida violada pela solução será encontrada, esse algoritmo será chamado uma heurística de separação.

Dada uma formulação para um problema qualquer, suponha que exista em sua correspondente região poliedral $\mathcal{R}$ uma classe $\mathcal{D}$ de exponencialmente muitas restrições. A medida que a dimensão do problema aumenta, a quantidade de restrições dessa classe aumenta explosivamente, tornando impraticável a resolução direta da relaxação linear daquela formulação. Alternativamente, considerando essa classe de restrições da formulação como desigualdades válidas, os algoritmos de *Branch and Cut* podem ser utilizados para tratar esse tipo de formulação. Isso é feito resolvendo em cada nó da árvore de enumeração uma relaxação linear contendo todas as restrições de $\mathcal{R}$ que não pertencem à $\mathcal{D}$ e um subconjunto reduzido das restrições contidas em $\mathcal{D}$. Em seguida, as restrições de $\mathcal{D}$ violadas pela solução da relaxação linear corrente são separadas através de um algoritmo de separação exato e incluídas na relaxação corrente. Esse processo continua até que não haja restrições em $\mathcal{D}$ violadas pela solução ótima da relaxação linear corrente.

Frequentemente, num dado nó da árvore de procura de um algoritmo *Branch and Cut*, um número crescente de planos de corte é demandado para obter ganhos cada vez mais reduzidos nos limitantes duais. Num caso extremo esses ganhos ficam estagnados embora desigualdades violadas continuam a ser identificados. Esse comportamento tende a reduzir significativamente a eficiência do algoritmo de *Branch and Cut*. Para atenuar seu efeito, diversas metodologias foram propostas (KOCH e MARTIN, 1998; LUCENA e RESENDE, 2004; PADBERG e RINALDI, 1991). Uma metodologia simples, largamente utilizada para lidar com esse problema, consiste em estabelecer um número máximo de iterações consecutivas (MAXITER) de

relaxações lineares (ou iterações do Método de Planos de Cortes) sem obter pelo menos uma melhora mínima (MINIMP) para o limitante dual. quando este limitante é atingido, o Método de Planos de Cortes é interrompido e uma ramificação é efetuada no nó em questão.

Nesta tese, foram desenvolvidos algoritmos do tipo *Branch and Cut* (NEMHAUSER e WOLSEY, 1988) com o objetivo de encontrar soluções exatas para ambas as variantes do PCD2CM. Para a variante 2-aresta conexa foram implementados algoritmos *Branch and Cut* baseados tanto na formulação (2.7) quanto na sua reformulação direcionada que utiliza um nó raiz artificial (2.37). Tais algoritmos são respectivamente denominados EUCUT e EDCUT.

Para a variante 2-vértice conexa do problema também implementamos algoritmos *Branch and Cut*. Estes se baseiam na formulação (2.40) e na reformulação direcionada (2.45) e são denominados respectivamente NUCUT e NDCUT.

Foram também implementadas versões dos algoritmos EUCUT e NUCUT que resultam da inserção das desigualdades válidas (2.50) nas formulações (2.7) e (2.40). Esses algoritmos são denominados respectivamente EUCUT+ e NUCUT+.

A seguir serão descritos em detalhes alguns dos aspectos mais importantes dos algoritmos *Branch and Cut* aqui implementados.

4.1 Algoritmos EUCUT e EUCUT$^+$

No nó raiz da árvore de enumeração do algoritmo EUCUT, um limitante inferior válido para (2.7) é obtido pela resolução de sua relaxação linear correspondente. Por outro lado, um limitante inferior computacionalmente mais barato para (2.8) é obtido restringindo as desigualdades de corte (2.4) para

$$\sum_{e \in \delta(i)} x_e \geq 2y_i, \qquad i \in V. \tag{4.6}$$

Seja $(\bar{x}, \bar{y})$ uma solução ótima associada ao limitante de (2.8) que acabamos de descrever e $\bar{G} = (\bar{V}, \bar{E})$ o seu correspondente grafo suporte. Dessa maneira, temos que $\bar{V} = \{i \in V : \bar{y}_i > 0\}$ e $\bar{E} = \{e \in E : \bar{x}_e > 0\}$. Se $(\bar{x}, \bar{y})$ não viola qualquer desigualdade de corte (2.4), ela é ótima para (2.8). Adicionalmente, se os valores $\bar{\mathbf{y}}$ forem todos inteiros, $(\bar{x}, \bar{y})$ será também uma solução ótima para (2.7). Caso contrário, a relaxação deve ser reforçada com desigualdades (2.4), violadas.

Para separar as desigualdades (2.4) é definida uma rede $\bar{N} = (\bar{V}, \bar{A})$, que se origina de $\bar{G}$. Esta rede é tal que, para cada aresta $e \in \bar{E}$, dois arcos são definidos em $\bar{A}$ com as mesmas extremidades de e e sentidos opostos. Além disso, uma capacidade $\overline{x}_e$ deve estar associada à cada um desses arcos. Dado $\bar{N}$, um corte de menor capacidade separando i e j deve ser identificado para todo $i, j \in \bar{V}$ tal que

$i \neq j$. Quando a capacidade do corte é menor que $2(\bar{y}_i + \bar{y}_j - 1)$, a desigualdade (2.4) a ele associada é violada.

Encontrar um corte de capacidade mínima é equivalente à identificar o fluxo máximo com origem em i e destino em j em $\bar{N}$ (FORD e FULKERSON, 1956). Para obter fluxos máximos utilizamos a implementação de CHERKASSKY e GOLDBERG (1997) para o algoritmo *Push-Relabel* de GOLDBERG e TARJAN (1988). Entretanto, devemos notar que as demandas computacionais associadas ao procedimento de separação que acabamos de descrever podem ser eventualmente reduzidas. Isso ocorre quando a desigualdade $y_i + y_j - 1 \leq 0$ é satisfeita. Neste caso, a desigualdade de corte (2.4) é naturalmente satisfeita para $i, j \in V$ e pode então ser desconsiderada durante o procedimento de separação. Além disso, se a quantidade de pares de vértice distintos $i, j \in \bar{V}$ satisfazendo $\bar{y}_i + \bar{y}_j > 1$ é maior que $|\bar{V}|$, a árvore de GOMORY e HU (1961), se torna então a opção de escolha para resolver nossos problemas de fluxo-máximo. Isso ocorre porque tal estrutura permite calcular os fluxos máximos entre todos os pares de vértices de $\bar{N}$ em tempo correspondente a $|V|$ vezes o tempo necessário para o cálculo de um único fluxo máximo. Tendo em mente que, no máximo, um problema de fluxo-máximo deve ser resolvido para cada vértice de V, a complexidade total do problema de separação para as desigualdades (2.4) é portanto $O(|V|^4)$. O Algoritmo 10 descreve um pseudo-código para o procedimento de separação das desigualdades (2.4).

Desigualdades violadas (2.4) devem ser adicionadas à relaxação de (2.7) que deve ser então resolvida novamente. Se a solução ótima assim obtida, $(\bar{x}, \bar{y})$, for inteira, esta será também ótima para (2.7). Caso contrário, o procedimento de separação deve ser repetido até que a solução obtida seja inteira, ou, alternativamente, esta seja fracionária mas nenhuma desigualdade violada exista em (2.4). Quando o último caso se aplica, devemos então recorrer à ramificação.

O resolvedor CPLEX 12.4 (CPLEX, 2013) foi utilizado para implementar o algoritmo EUCUT. Entretanto, os testes de pré-processamento, heurísticas e geradores de planos de corte automáticos que oferece foram desativas. O procedimento de ramificação foi conduzido sobre as variáveis $\mathbf{y}$, deixando a administração da árvore de enumeração para o resolvedor. Para cada nó da árvore de enumeração, o Problema de Programação Linear correspondente foi reforçado como descrito acima.

No algoritmo EUCUT, a segunda heurística construtiva e o procedimento para redução de cardinalidade propostos no Capítulo 3 foram utilizados para obter um limitante superior inicial para a árvore de enumeração.

Nesta tese, implementamos também uma versão do algoritmo EUCUT, denominada EUCUT$^+$, onde a região poliedral $\mathcal{R}_1$ é reforçada com a inserção das desigualdades (2.50). O algoritmo EUCUT$^+$ separa essas desigualdades para cada par de

Dados: Uma solução ótima $(\bar{x}, \bar{y})$ para a relaxação linear corrente e o seu grafo suporte $\bar{G} = (\bar{V}, \bar{E})$ correspondente.
Resultado: Um conjunto de desigualdades violadas a serem inseridas no PL.
Construir uma rede $\bar{N} = (\bar{V}, \bar{A})$ tal que $\bar{A} = \{(i,j),(j,i) \ : \ \{i,j\} \in \bar{E}\}$.
$K \leftarrow$ Quantidade de pares $i, j \in \bar{V}$ tais que $i < j$ e $\bar{y}_i + \bar{y}_j > 1$.
se $K \leq |V|$ **então**
  para cada $i \in \bar{V}$ **faça**
    para cada $j \in \bar{V}$, $i < j$ **faça**
      se $\bar{y}_i + \bar{y}_j > 1$ **então**
        $f \leftarrow$ valor do fluxo-máximo com origem em i e destino em j na rede $\bar{N}$.
        se $f < 2(\bar{y}_i + \bar{y}_j - 1)$ **então**
          Detectar o corte $\delta(S)$ tal que $\sum_{e \in \delta(S)} \bar{x}_e = f$.
          Inserir o corte violado $\sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1)$ na relaxação corrente.
        fim
      fim
    fim
  fim
senão
  Calcular a Árvore de Gomory-Hu para a rede $\bar{N}$.
  para cada $i \in \bar{V}$ **faça**
    para cada $j \in \bar{V}$, $i < j$ **faça**
      se $\bar{y}_i + \bar{y}_j > 1$ **então**
        $f \leftarrow$ valor do fluxo-máximo obtido com origem em i e destino em j encontrado pela Árvore de Gomory-Hu.
        se $f < 2(\bar{y}_i + \bar{y}_j - 1)$ **então**
          Detectar o corte $\delta(S)$ tal que $\sum_{e \in \delta(S)} \bar{x}_e = f$.
          Inserir o corte violado $\sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1)$ na relaxação corrente.
        fim
      fim
    fim
  fim
fim

Algoritmo 10: Separação das desigualdades 2.4.

vértices $i, j \in \bar{V}$ distintos, onde $\Gamma_i \cap \Gamma_j = \emptyset$. Para isso, um grafo suporte $\bar{G}_{ij}$ é obtido após contrair todos os vértices pertencentes à $\Gamma_i \cap \bar{V}$ (resp. $\Gamma_j \cap \bar{V}$) em um super-vértice i (resp. j). Arestas múltiplas que possam eventualmente resultar desta operação, são também contraídas em super-arestas correspondentes. Neste caso, a capacidade dessa super-aresta será igual à soma das capacidades das arestas que lhe deram origem. Tendo feito isso, uma rede $\bar{N}_{ij}$ é então construída a partir de $\bar{G}_{ij}$, tal como descrito anteriormente para $\bar{G}$ e $\bar{N}$. Um fluxo máximo com origem em i e destino em j deve ser então computado sobre $\bar{N}_{ij}$. Quando este for

menor que 2, a desigualdade (2.50) que lhe corresponde é então violada e deve ser utilizada para restringir a região de viabilidade corrente. Como descrito anteriormente, parâmetros MINIMP e MAXITER estão associados a nosso procedimento de fortalecimento da relaxação e têm por objetivo evitar a estagnação do algoritmo em um nó da árvore de enumeração. O valor adotado para esses parâmetros foram de 0,001 para MINIMP e 10 iterações para MAXITER. O procedimento de separação descrito acima é expresso pelo pseudo-código correspondente ao Algoritmo 11.

Dados: Uma solução ótima $(\bar{x}, \bar{y})$ para a relaxação linear corrente e o seu grafo suporte $\bar{G} = (\bar{V}, \bar{E})$ correspondente.
Resultado: Um conjunto de desigualdades violadas a serem inseridas no LP corrente.
para cada $i \in V$ **faça**
  para cada $j \in V$, $i \neq j$ **faça**
    Construir o grafo $H_{ij} = (V_{ij}, E_{ij})$ a partir da contração dos vértices em Γ_i e Γ_j nos super-vértices a e b.
    Construir uma rede $\bar{N}$ a partir de H_{ij}.
    $f \leftarrow$ valor do fluxo-máximo com origem em a e destino em b na rede $\bar{N}$.
    se $f < 2$ **então**
      Detectar o corte tal que $\sum_{e \in delta(S))} \bar{x}_e = f$.
      Inserir o corte violado $\sum_{e \in \delta(S)} x_e \geq 2$ no LP.
    fim
  fim
fim

Algoritmo 11: Separação das desigualdades (2.50).

4.2 Algoritmo EDCUT

Para o algoritmo EDCUT, no nó raiz da árvore de enumeração, um limitante inferior válido para a formulação (2.37) e dado pela resolução de sua relaxação linear (2.38). Por outro lado, um limitante inferior computacionalmente mais barato de (2.38) é obtido substituindo-se, em uma primeira etapa, as desigualdades (2.28) e (2.29) pelas desigualdades

$$\sum_{a \in \delta^+(i)} h_a \geq y_i,\ i \in V \tag{4.7}$$

e

$$\sum_{a \in \delta^-(i)} h_a \geq y_i,\ i \in V. \tag{4.8}$$

Seja $(\bar{h}, \bar{z}, \bar{y})$ uma solução ótima para tal relaxação de (2.38) e $\bar{D}_r = (\bar{V}_r, \bar{A} \cup \bar{R})$ o seu grafo suporte. Dessa maneira, temos que $\bar{V}_r = \{i \in V : \bar{y}_i > 0\} \cup \{r\}$, $\bar{A} = \{a \in A : \bar{h}_a > 0\}$, e $\bar{R} = \{(r,i),(i,r) : \bar{z}_i > 0\}$. Se $(\bar{h}, \bar{z}, \bar{y})$ violar nenhuma das desigualdades de corte (2.28) e (2.29), este é então uma solução ótima para (2.38). Além disso, se todos os valores de $\bar{y}$ e $\bar{z}$ forem inteiros, esta será também uma solução ótima para a formulação (2.37). Caso contrário, a relaxação deve ser reforçada adicionando as desigualdades violadas encontradas em (2.28) e (2.29).

O problema de separação para as desigualdades (2.28) e (2.29) é resolvido como uma sequência de problemas de fluxo-máximo definidos diretamente sober o digrafo $\bar{D}_r = (\bar{V}_r, \bar{A} \cup \bar{R})$. Neste processo, capacidades h_a e z_i são associadas à cada arco correspondente de $\bar{A}$ e $\bar{R}$, respectivamente. Por exemplo, para todo $i \in V_r$, devemos encontrar o fluxo máximo com origem em i e destino em r e com origem em r e destino em i, onde $r \in S_i$ e $i \in \bar{V}_i \setminus S_i$, para o último caso. Quando (2.28) (respectivamente (2.29)) é violado para $S = S_r$ (respectivamente $S = S_i$), uma desigualdade violada (2.28) (respectivamente (2.29)) é identificada. Como é necessário calcular o fluxo máximo duas vezes para cada vértices $i \in V$, tal que $y_i > 0$, o esforço computacional da separação das desigualdades (2.28) e (2.29) será igual à duas vezes o esforço computacional para resolver um único problema de fluxo-máximo no grafo $\bar{D}_r$, ou seja, $O(|V|^4)$. O procedimento de separação descrito acima é ilustrado pelo pseudo-código apresentado no Algoritmo 12.

Dados: Uma solução ótima $(\bar{h}, \bar{z}, \bar{y})$ para a relaxação linear corrente e o seu grafo suporte $\bar{D}_r = (\bar{V}_r, \bar{A} \cup \bar{R})$ correspondente.
Resultado: Um conjunto de desigualdades violadas a serem inseridas no PL.
para cada $i \in V_r$ *e* $i \neq r$ **faça**
  $f \leftarrow$ valor do fluxo-máximo com origem em r e destino em i no digrafo $\bar{D}_r$.
  se $f < \bar{y}_i$ **então**
    Detectar o corte $\delta^+(S)$ tal que $\sum_{a \in \delta^+(S)} h_a + \sum_{j \in \Gamma_{min} \setminus S} z_j = f$.
    Inserir o corte violado $\sum_{a \in \delta^+(S)} h_a + \sum_{j \in \Gamma_{min} \setminus S} z_j \geq y_i$ no PL.
  fim
  $f \leftarrow$ valor do fluxo-máximo com origem em i e destino em r no digrafo $\bar{D}_r$.
  se $f < \bar{y}_i$ **então**
    Detectar o corte $\delta^-(S)$ tal que $\sum_{a \in \delta^-(S)} h_a + \sum_{j \in \Gamma_{min} \setminus S} z_j = f$.
    Inserir o corte violado $\sum_{a \in \delta^-(S)} h_a + \sum_{j \in \Gamma_{min} \setminus S} z_j \geq y_i$ no PL.
  fim
fim

Algoritmo 12: Separação das desigualdades (2.28) e (2.29).

Sob os mesmos comentários gerais feitos para o algoritmo EUCUT, o algoritmo EDCUT também foi implementado utilizando o resolvedor CPLEX 12.4. A ramificação, nesse caso, foi conduzida sobre as variáveis $(\mathbf{y}, \mathbf{z})$. Em todos os nós da árvore de enumeração, relaxações lineares foram obtidas como descrito acima.

Da mesma forma como foi feito para o algoritmo EUCUT, o algoritmo EDCUT utiliza a segunda heurística construtiva, descrita na Seção 3.3, em conjunto com o procedimento para redução de cardinalidade, descrito na Seção 3.4, para obter um limitante superior inicial para a árvore de enumeração.

4.3 Algoritmos NUCUT e NUCUT$^+$

Como descrito anteriormente para o algoritmo EUCUT, no nó raiz da árvore de enumeração do algoritmo NUCUT, um limitante inferior válido para (2.40) é obtido pela resolução de sua relaxação linear (2.41). Entretanto, como antes, um limitante inferior computacionalmente mais barato de (2.41) é obtido substituindo-se, em uma primeira etapa, as desigualdades (2.4) e (2.39) por

$$\sum_{e \in \delta(i)} x_e \geq 2y_i, \qquad i \in V. \tag{4.9}$$

Seja $(\bar{x}, \bar{y})$ uma solução ótima para tal relaxação de (2.41) e $\bar{G} = (\bar{V}, \bar{E})$, como definido na Seção 4.1, o seu grafo suporte correspondente. Se $(\bar{x}, \bar{y})$ não violar qualquer desigualdade de corte (2.4) e (2.39), será também uma solução ótima para (2.41). Ademais, se os valores de $\mathbf{y}$ para esta solução forem inteiros, ela será também uma solução ótima para a formulação (2.40). Caso contrário, a relaxação deve ser reforçada com as desigualdades violadas encontradas em (2.4) e (2.39).

O procedimento de separação das desigualdades (2.4) e (2.39) foi implementado através de uma adaptação do procedimento sugerido em GRÖTSCHEL *et al.* (1995). Isso é feito introduzindo uma rede auxiliar $N^+ = (V^+, A^+)$, construída a partir do grafo $\bar{G} = (\bar{V}, \bar{E})$, como explicado a seguir. Para todo $u \in \bar{V}$, dois vértices u_1 and u_2 são introduzidos em V^+, juntamente com um arco (u^1, u^2) para A^+. A capacidade desses arcos é fixada em uma unidade. Alem disso, para cada aresta $e = \{u, v\} \in \bar{E}$, dois arcos (u^2, v^1) e (v^2, u^1) são também introduzidos em A^+, cada um deles com uma capacidade igual a $\bar{x}_e$.

Devemos então calcular o corte de menor capacidade separando os vértice s^2 e t^1, em N^+, para todo par de vértices s^2 e t^1 distintos. Seja $\delta^+(S')$ o corte mínimo associado a um dado par s^2 e t^1, e f a sua capacidade. Devemos então identificar os seguintes conjuntos de vértices: $Z = \{u \in V^+ : (u^1, u^2) \in \delta^+(S')\}$ e $S = \{u \in V^+ : u^1, u^2 \in S'\}$. Feito isso, obtemos $\delta_{\{G \setminus Z\}}(S) = \{\{u, v\} \in E^+ : (v^2, u^1) \in \delta^+(S')\}$. Se $Z = \emptyset$ e $f < 2(\bar{y}_s + \bar{y}_t - 1)$ então a desigualdade $\sum_{e \in \delta_G(S)} x_e \geq 2(y_s + y_t - 1)$ é violada. Por outro lado, se $Z \neq \emptyset$ e $f < \bar{y}_s + \bar{y}_t$ então a desigualdade $\sum_{e \in \delta_{\{G \setminus Z\}}(S)} x_e \geq y_s + y_t - 1$ é violada. O procedimento de separação para as desigualdades (2.4) e (2.39) é descrito por um pseudo-código, no Algoritmo 13.

Denotamos por NUCUT o algoritmo *Branch and Cut* que acabamos de descrever.

Como feito feito anteriormente, este também foi implementado utilizando o resolvedor CPLEX 12.4. O procedimento de ramificação, foi conduzido sobre a variável $\mathbf{y}$. Em todos os nós da árvore de enumeração, relaxações lineares foram obtidas como descrito acima.

Um limitante superior inicial para o nó raiz da árvore de enumeração foi obtido utilizando a solução primal encontrada pela heurística da Seção 3.3 e refinado pela aplicação do procedimento de redução de cardinalidade da Seção 3.4.

Dados: Uma solução ótima $(\bar{x}, \bar{y})$ para a relaxação linear corrente e o seu grafo suporte $\bar{G} = (\bar{V}, \bar{E})$ correspondente.
Resultado: Um conjunto de desigualdades violadas a serem inseridas no LP corrente.
Uma rede $N^+ = (V^+, A^+)$ é construída a partir de $\bar{G}$.
para cada $s^2 \in V^+$ **faça**
  para cada $t^1 \in V^+$, $s^1 \neq t^1$ **faça**
    se $\bar{y}_i + \bar{y}_j > 1$ **então**
      $f \leftarrow$ valor do fluxo-máximo com origem em s^2 e destino em t^1 na rede N^+.
      Detectar o corte $\delta^+(S')$ tal que $\sum_{a \in \delta^+(S')} \bar{x}_a = f$.
      $Z = \{u \in V \ : \ (u_1, u_2) \in \delta^+(S')\}$
      $S = \{u \in V \ : \ u_1, u_2 \in S'\}$
      se $Z = \emptyset$ **então**
        se $f < 2(\bar{y}_i + \bar{y}_j - 1)$ **então**
          $\delta(S) = \{\{u, v\} \in E \ : \ (v_2, u_1) \in \delta^+(S')\}$
          Inserir o corte violado $\sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1)$ no LP.
        fim
      senão
        se $f < \bar{y}_i + \bar{y}_j$ **então**
          $\delta_{G \setminus Z}(S) = \{\{u, v\} \in E \ : \ (v_2, u_1) \in \delta^+(S')\}$
          Inserir o corte violado $\sum_{e \in \delta_{G \setminus Z}(S)} x_e \geq y_i + y_j - 1$ no LP.
        fim
      fim
    fim
  fim
fim

Algoritmo 13: Separação das desigualdades 2.4 e 2.39.

Implementamos também uma versão do algoritmo NUCUT, denotada por NUCUT$^+$, onde a região de viabilidade é definida pela interseção da região poliedral $\mathcal{R}_6$ com as desigualdades (2.50). Neste caso, a separação das desigualdades (2.50) é feita utilizando o mesmo procedimento usado no algoritmo EUCUT$^+$ definido na Seção 4.1. Além disso, também foi adotado um valor de 10 iterações para o parâmetro MAXITER e 0,001 para o parâmetro MINIMP como uma medida para evitar a estagnação do algoritmo.

4.4 Algoritmo NDCUT

Para o algoritmo NDCUT, no nó raiz da árvore de enumeração, um limitante inferior válido para a formulação (2.45) é dado pela resolução de sua relaxação linear (2.46). Por outro lado, um limitante inferior computacionalmente mais barato que (2.46) é obtido desconsiderando-se inicialmente as desigualdades (2.28), (2.29) e (2.42) e utilizando as seguintes desigualdades:

$$\sum_{a \in \delta^+(i)} h_a \geq y_i,\ i \in V \tag{4.10}$$

e

$$\sum_{a \in \delta^-(i)} h_a \geq y_i,\ i \in V. \tag{4.11}$$

Seja $(\bar{h}, \bar{z}, \bar{y})$ uma solução ótima para a relaxação de (2.46) assim obtida e $\bar{D}_r = (\bar{V}_r, \bar{A} \cup \bar{R})$ o seu grafo suporte correspondente. Dessa maneira, temos que $\bar{V}_r = \{i \in V : \bar{y}_i > 0\} \cup \{r\}$, $\bar{A} = \{a \in A : \bar{h}_a > 0\}$, e $\bar{R} = \{(r,i),(i,r) : \bar{z}_i > 0\}$. Se $(\bar{h}, \bar{z}, \bar{y})$ não violar nenhuma das desigualdades de corte (2.28), (2.29) e (2.42), esta é uma solução ótima para (2.46). Além disso, se todos os valores de $\bar{\mathbf{y}}$ e $\bar{\mathbf{z}}$ forem inteiros, ela também será uma solução ótima para a formulação (2.45). Caso contrário, a relaxação deve ser reforçada com as desigualdades violadas encontradas em (2.28), (2.29) e (2.42).

O problema de separação para as desigualdades (2.28) e (2.29) é resolvido utilizando o mesmo procedimento de separação descrito anteriormente para o algoritmo EDCUT. Para as desigualdades (2.42) o problema é resolvido como descrevemos a seguir. Tomando $B = \{i \in \bar{V}_r \cap \Gamma_{min} : \bar{z}_i > 0\}$, inicialmente construímos, para todo $k \in \bar{V}_r \setminus \{r\}$ e $k \notin B$, uma rede $\bar{N}_k = (\bar{V}_r \setminus \{k\}, (\bar{A} \cup \bar{R}) \setminus (\delta^+(k) \cup \delta^-(k)))$, que se origina de $\bar{D}_r$. A capacidade associada a cada arco (i,j), onde $i,j \neq r$, deve ser igual à soma $\overline{h}_{(i,j)} + \overline{h}_{(j,i)}$. Por outro lado, para os arcos $(r,i),(i,r) \in R$ a capacidade deve ser fixada em $\overline{z}_i$. Feito isso, para cada rede construída, identificamos o corte de menor capacidade entre r e i, onde $i \in \bar{V}_r \setminus \{k,r\}$. Quando a capacidade desse corte é menor que $\bar{y}_i$, sua desigualdade (2.42) correspondente é então violada. Para soluções onde os valores de $\bar{\mathbf{z}}$ são fracionários, é importante salientar que o procedimento de separação descrito acima não é exato, uma vez que desconsideramos a separação de todas as desigualdades envolvendo $k \in B$. No entanto, se todos os valores de $\bar{\mathbf{z}}$ forem inteiros, temos que B possuí um único vértice e portanto o procedimento de separação é exato. Esse procedimento de separação é representado pelo pseudo-código 14.

Tal como nos algoritmos apresentados anteriormente, o algoritmo NDCUT tam-

bém foi implementado utilizando o resolvedor CPLEX 12.4. O procedimento de ramificação, como feito em EDCUT, foi conduzido sobre as variáveis $(\mathbf{y}, \mathbf{z})$. Em todos os nós da árvore de enumeração, relaxações lineares foram resolvidas como descrito acima. O procedimento de separação para as desigualdades (2.42) é descrito por um pseudo-código, no Algoritmo 14.

A heurística construtiva descrita na Seção 3.3, foi utilizada para gerar soluções viáveis para o problema. Em conjunto com esta, utilizamos também o procedimento de redução de cardinalidade, descritos na Seção 3.4, para tentar melhorar a qualidade das soluções obtidas. O uso desses procedimentos se restringiu ao nó zero da árvore de enumeração.

Dados: Uma solução ótima $(\bar{h}, \bar{z}, \bar{y})$ para a relaxação linear corrente e o seu grafo suporte $\bar{D}_r = (\bar{V}_r, \bar{A} \cup \bar{R})$ correspondente.
Resultado: Um conjunto de desigualdades violadas a serem inseridas no PL.
Identificar o conjunto $B = \{i \in \bar{V}_r \cap \Gamma_{min} \ : \ \bar{z}_i > 0\}$
para cada $k \in V_r$, $k \neq r$ *e* $k \notin B$ **faça**
  Construir uma rede $\bar{N}_k = (\bar{V}_r \setminus \{k\}, (\bar{A} \cup \bar{R}) \setminus (\delta^+(k) \cup \delta^-(k)))$ **para cada** $i \in \bar{V}_r \setminus \{k, r\}$ **faça**
    $f \leftarrow$ valor do fluxo-máximo com origem em r e destino em i na rede $\bar{N}_k$.
    se $f < \bar{y}_i$ **então**
      Detectar o corte $\delta^-(S)$ em $\bar{N}_k$ tal que $\sum_{a \in \delta^-_{D_{V \setminus \{k\}}}(S)} h_a + \sum_{a \in \delta^+_{D_{V \setminus \{k\}}}(S)} h_a + \sum_{i \in \Gamma_{min} \setminus S} z_i = f$.
      Inserir o corte violado $\sum_{a \in \delta^-_{D_{V \setminus \{k\}}}(S)} h_a + \sum_{a \in \delta^+_{D_{V \setminus \{k\}}}(S)} h_a + \sum_{i \in \Gamma_{min} \setminus S} z_i \geq y_i$ no PL.
    fim
  fim
fim

Algoritmo 14: Separação das desigualdades (2.42).

Capítulo 5

Experimentos Computacionais

Neste capítulo apresentaremos os resultados numéricos obtidos com os experimentos computacionais conduzidos neste trabalho. Estes têm como objetivo comparar os diferentes algoritmos que implementamos para as duas versões do PCD2CM. Especificamente, vamos comparar os algoritmos EUCUT e EDCUT, implementados para resolver a variante 2-aresta conexa do problema. Da mesma forma, vamos também comparar os algoritmos NUCUT e NDCUT, implementados para resolver a variante 2-vértice conexa do PCD2CM. Além disso, os experimentos servem para avaliar os benefícios obtidos ao reforçar as formulações propostas para ambas as variantes do problema com as desigualdades válidas (2.50). Isso é feito ao analisar o desempenho dos algoritmos EUCUT$^+$, para a variante 2-aresta conexa, e NUCUT$^+$, para a variante 2-vértice conexa.

Os algoritmos propostos nesta tese foram implementados em linguagem C++, utilizando o compilador g++, com chave de otimização -O3 ativada. Todos os testes computacionais foram realizados em uma máquina com processador Intel Xeon 3.00 GHz e 4 Gbits de memória RAM, sob o sistema operacional Ubuntu 12.04. Apesar da máquina utilizada possuir múltiplos núcleos de processamento, apenas um destes foi utilizado nos experimentos. Para cada instância testada, um tempo limite de execução de 7200 segundos (2 horas) de CPU foi imposto a cada algoritmo considerado.

As instâncias utilizadas nos experimentos estão divididas em dois conjuntos, ambos introduzidos nesta tese. O primeiro está associado a grafos $G = (V, E)$ gerados pseudo-aleatoriamente com $|V| \in \{30, 50, 70, 100, 120, 150, 200\}$ e porcentagens de densidade $\{5, 10, 25, 50, 70\}$, com exceção da instância envolvendo 30 vértices e densidade 5%, que não foi possível gerar pelo nosso procedimento. Este, para garantir a satisfação da exigência de 2-{aresta, vértice} conexidade das instâncias, utiliza grafos inicializados por um ciclo Hamiltoniano onde os vértices são visitados na ordem $1, \ldots, |V|$. Feito isso, arestas adicionais são geradas pseudo-aleatoriamente até que a densidade exigida para o grafo da definição da instância seja atingida.

As instâncias para o segundo conjunto foram construídas com o objetivo de produzirem soluções diferentes para cada uma das variantes do problema. Construímos essas instâncias como descrevemos a seguir. Assuma, como anteriormente, que um grafo $G = (V, E)$ com densidade d será gerado pseudo-aleatoriamente. O conjunto de vértices $|V|$ é inicialmente particionado em $A \cup B \cup \{i_{|V|-1}, i_{|V|}\}$, onde $|A| = |B|$. O conjunto de arestas E é então inicializado com apenas dois caminhos, $P_A = i_1, \ldots, i_{|A|}$ e $P_B = i_{|A|+1}, \ldots, i_{|A|+|B|}$, envolvendo respectivamente todos os vértices em A e em B. Além disso, arestas $\{i_1, i_{|V|-1}\}$, $\{i_{|A|}, i_{|V|-1}\}$, $\{i_{|A|}, i_{|V|}\}$, $\{i_{|A|+1}, i_{|V|-1}\}$, $\{i_{|A|+1}, i_{|V|}\}$, e $\{i_{|A|+|B|}, i_{|V|-1}\}$, são também introduzidas em G. Por fim, são adicionadas tantas arestas quantas forem necessárias para que o grafo G atinja a densidade exigida. Para a variante 2-vértice conexa do problema, soluções para essas instâncias devem obrigatoriamente conter os vértices $i_{|V|-1}$ e $i_{|V|}$. Enquanto, para a variante 2-aresta conexa, basta que um deles faça parte da solução. Instâncias com $|V| \in \{30, 50, 70, 100, 120, 150, 200\}$ com porcentagem da densidade $\{5, 10, 25\}$, com exceção da instância com 30 vértices e densidade 5%, foram geradas como acabamos de descrever.

5.1 Experimentos Para a Variante 2-Aresta Conexa

Os resultados obtidos para os algoritmos EUCUT e EDCUT são apresentados nas Tabelas 5.1 e 5.2. A primeira está associada ao primeiro conjunto de instâncias de teste enquanto a outra se refere ao segundo conjunto. As tabelas indicam, para cada instância testada, a cardinalidade do conjunto de vértices ($|V|$), a densidade do grafo $G = (V, E)$ (dens.%), o valor da solução ótima (SO) ou " - ", quando este é desconhecido, o valor da solução primal obtida pela heurística primal (LS), os valores obtidos pelos algoritmos EUCUT e EDCUT para as relaxações lineares no nó zero da árvore de enumeração (LI), o número de nós da árvore de enumeração até o final da execução, e o tempo de CPU, em segundos, ou o valor do GAP(%), caso o limite de 7200 segundos seja atingido.

Comparando os resultados apresentados obtidos pelos algoritmos EUCUT e EDCUT para o primeiro conjunto de instâncias, observamos que o primeiro foi capaz de encontrar a solução ótima e provar a sua otimalidade para todas as instâncias com até 150 vértices e também para duas instâncias com 200 vértices e densidades de 5% e 70%. No entanto, EUCUT falhou em resolver à otimalidade três instâncias contendo 200 vértices e densidades 10%, 25% e 50%. Já o algoritmo EDCUT falhou para 12 instâncias, sendo bem sucedido em encontrar soluções comprovadamente ótimas para todas as instâncias com até 100 vértices. Além disso, obteve soluções comprovadamente ótimas para as instâncias com 120 vértices e densidades 5% e 70% e aquelas com 150 vértices e densidade 70%. Para as instâncias onde ambos os

				EUCUT			EDCUT		
$\|V\|$	dens.(%)	SO	LS	LI	NN	$t(s)$	LI	NN	$t(s)$
30	10	19	20	18.000	3	0.211	18.000	5	0.089
	25	6	7	5.253	9	0.065	5.253	13	0.085
	50	3	3	2.030	1	0.04	2.030	1	0.04
	70	3	3	1.496	13	0.092	1.496	14	0.149
50	5	15	18	14.000	11	0.091	14.000	5	0.097
	10	9	10	7.830	11	0.126	7.830	10	0.144
	25	6	6	4.391	20	1.363	4.391	23	0.311
	50	3	3	2.040	1	0.33	2.040	1	0.32
	70	3	3	1.465	22	4.034	1.465	5	0.373
70	5	31	34	30.571	4	0.196	30.571	3	0.183
	10	14	16	11.832	114	1.097	11.832	42	1.574
	25	7	7	4.263	542	4.19	4.263	514	16.63
	50	4	4	2.069	125	1.795	2.069	151	9.873
	70	3	3	1.453	44	1.076	1.453	63	2.189
100	5	30	34	27.667	33	0.541	27.667	87	21.282
	10	15	17	11.122	1716	268.814	11.122	2240	3094.24
	25	7	7	4.115	2465	59.962	4.115	1877	761.289
	50	3	4	2.042	99	10.429	2.042	204	682.033
	70	3	3	1.456	156	10.537	1.456	164	20.69
120	5	29	33	25.211	718	6.257	25.211	1031	1425.54
	10	15	16	10.650	7010	258.667	10.650	1348	14.94%
	25	7	7	4.115	2561	70.925	4.115	81	36.78%
	50	4	4	2.054	205	45.525	2.054	37	45.89%
	70	3	3	1.438	170	14.175	1.438	198	184.247
150	5	27	32	23.758	1509	102.832	23.758	2471	6.70%
	10	15	17	10.862	54772	4797.17	10.862	985	33.21%
	25	7	8	4.113	55590	2766.03	4.113	27	51.24%
	50	4	5	2.014	5656	481.286	2.014	236	57.40%
	70	3	3	1.433	256	77.828	1.433	289	174.364
200	5	28	33	22.636	57703	3424.79	22.636	33	25.76%
	10	-	19	10.606	40880	32.07%	10.606	72	42.52%
	25	-	8	4.097	61750	39.18%	4.097	24	52.13%
	50	-	5	2.014	11751	40%	2.014	226	51%
	70	3	3	1.434	343	392.474	1.434	162	51.17%

Tabela 5.1: Comparação entre EUCUT e EDCUT para o primeiro conjunto de instâncias de teste.

algoritmos falharam, o GAP entre os melhores limitantes obtidos durante a árvore de enumeração foi maior para o algoritmo EDCUT.

Para o segundo conjunto de instâncias, apenas aquelas com até 120 vértices e densidade 10% e aquela com 150 vértices e densidade 5% tiveram a sua otimalidade provada pelo algoritmo EUCUT. Nenhuma instância com 200 vértices teve a sua otimalidade provada. No total o algoritmo EUCUT falhou em provar a otimalidade para 6 instâncias desse conjunto no limite de tempo imposto. Por outro lado, o

				EUCUT			EDCUT		
$\|V\|$	dens.(%)	SO	LS	LI	NN	$t(s)$	LI	NN	$t(s)$
30	10	22	22	21.000	1	0.040	20.000	3	0.062
	25	8	8	5.000	16	0.158	5.000	35	0.489
50	5	40	41	40.000	1	0.230	39.714	2	0.033
	10	17	18	14.513	18	1.361	14.492	14	1.633
	25	8	8	5.245	11	1.061	5.318	14	3.925
70	5	32	34	31.400	7	2.539	31.400	4	1.054
	10	17	18	12.303	109	16.734	12.881	81	9.998
	25	8	9	4.907	94	72.742	4.897	98	48.219
100	5	30	36	28.465	34	20.438	28.466	68	41.087
	10	16	19	11.550	339	72.627	11.765	290	1273.24
	25	8	9	5.500	94	1903.270	5.520	4	34.73%
120	5	29	33	24.882	263	245.828	25.050	401	880.186
	10	17	19	11.664	3440	1718.290	11.733	4	37.50%
	25	10	11	5.555	3	45.27%	5.579	4	45.32%
150	5	30	35	24.340	2856	1855.56	24.494	66	28.56%
	10	-	22	12.212	1599	31.51%	12.295	4	43.90%
	25	-	10	5.623	3	38.05%	5.623	3	38.05%
200	5	-	35	23.367	662	23.02%	23.450	4	32.45%
	10	-	20	11.862	375	36.57%	11.864	4	39.45%
	25	-	10	5.646	4	43.54%	5.646	1	43.54%

Tabela 5.2: Comparação entre EUCUT e EDCUT para o segundo conjunto de instâncias de teste.

algoritmo EDCUT conseguiu encontrar soluções comprovadamente ótimas apenas para as instâncias com até 100 vértices e densidade 5%, falhando para 9 instâncias. Nenhuma instância com 150 e 200 vértices teve a sua otimalidade provada pelo algoritmo EDCUT. Para todas as instâncias do segundo conjunto, onde ambos os algoritmos falharam, o algoritmo EDCUT teve pior desempenho em termos do GAP de dualidade observado ao interrompermos a execução.

Para o primeiro conjunto de instâncias testadas, as relaxações lineares das formulações associadas a cada algoritmo tiveram sempre o mesmo valor. Para o segundo conjunto, uma pequena diferença entre esses valores pode ser notada. No entanto, tal diferença não é significativa, sendo de aproximadamente 0.58 unidades, a maior diferença encontrada.

Considerando as instâncias onde ambos os algoritmos foram bem sucedidos, o tempo de CPU médio consumido pelo algoritmo EDCUT foi de 290,716 segundos para o primeiro conjunto de instâncias e de 205,448 segundos para o segundo. Por outro lado, para o algoritmo EUCUT, os tempos correspondentes foram de 21,057 segundos para o primeiro conjunto de instâncias e de 39,414 segundos para o segundo.

Fazendo a média da razão entre o número de nós explorados na árvore de enumeração e o tempo de CPU gasto por cada algoritmo para resolver cada instância,

temos que a razão para o algoritmo EUCUT é de aproximadamente 41 e 11 nós por segundo para as instâncias do primeiro e segundo conjuntos, respectivamente, enquanto para o algoritmo EDCUT é de aproximadamente 19 e 8 nós por segundo para as instâncias do primeiro e segundo conjuntos, respectivamente. Com isso, podemos inferir que a resolução da relaxação linear em cada nó do algoritmo EUCUT é em média mais rápida que para o algoritmo EDCUT.

Pelas informações obtidas em nossos experimentos computacionais, podemos concluir que o algoritmo EUCUT tem melhor desempenho que o algoritmo EDCUT.

A seguir, será feita uma comparação entre os resultados obtidos pelos algoritmos EUCUT e EUCUT$^+$. Para o segundo algoritmo, a mesma formulação (2.7) utilizada pelo primeiro, é reforçada com as desigualdades válidas (2.50).

A Tabela 5.3 diz respeito ao primeiro conjunto de instâncias de teste enquanto a Tabela 5.4 está associada ao segundo. Essas tabelas estão estruturadas da mesma forma como descrevemos as Tabelas 5.1 e 5.2. Da mesma maneira, um tempo limite de 7200 segundos de CPU foi também imposto para a execução de cada instância testada.

Os resultados para o primeiro conjunto de instâncias mostram que o algoritmo EUCUT$^+$, além de não conseguir obter certificados de otimalidade para aquelas instâncias onde o algoritmo EUCUT falha, também não conseguiu obter tais certificados para 5 instâncias onde EUCUT obteve êxito. No entanto, para as instâncias onde ambos falharam, o algoritmo EUCUT$^+$ obteve um menor GAP entre os melhores limitantes obtidos durante a árvore de enumeração. Já para o segundo conjunto de instâncias testadas, vide a Tabela 5.4, os resultados obtidos mostram que, além das instâncias onde ambos falharam, o algoritmo EUCUT conseguiu obter um certificado de otimalidade para a instância com 150 vértices e densidade 5% e falhou para a instância com 120 vértice e densidade 25%. Para o algoritmo EUCUT$^+$, o inverso ocorre em relação a essas duas instâncias.

Em relação à qualidade dos limitantes inferiores obtidos, a incorporação das desigualdades válidas (2.50) à formulação (2.7) não resultou em limitantes mais fortes, para o primeiro conjunto de instâncias testadas. No entanto, para o segundo conjunto, limitantes inferiores mais fortes foram obtidos, sendo a maior diferença observada a favor de EUCUT$^+$, igual a 2,36 unidades, aproximadamente 32,44%, e, na média, sendo superior em 1,42 unidade, cerca de 12,98%.

Para as instâncias que ambos os algoritmos conseguiram resolver, o tempo computacional exigido pelo algoritmo EUCUT$^+$ foi maior que aquele exigido pelo algoritmo EUCUT. Em detalhes, o algoritmo EUCUT consumiu, em média, 47,33 e 311,93 segundos de CPU, respectivamente para o primeiro e o segundo conjuntos de instâncias testadas. Por outro lado, o algoritmo EUCUT$^+$ consumiu 360,52 e 260,50 segundos, em média, nos dois casos.

				EUCUT			EUCUT$^+$		
$\|V\|$	dens.(%)	SO	LS	LI	NN	$t(s)$	LI	NN	$t(s)$
30	10	19	20	18.000	3	0.211	18.000	3	0.178
	25	6	7	5.253	9	0.065	5.257	8	0.15
	50	3	3	2.030	1	0.04	2.030	1	0.04
	70	3	3	1.496	13	0.092	1.496	13	0.108
50	5	15	18	14.000	11	0.091	14.000	13	1.391
	10	9	10	7.830	11	0.126	7.852	10	1.685
	25	6	6	4.391	20	1.363	4.391	4	0.592
	50	3	3	2.040	1	0.33	2.040	1	0.33
	70	3	3	1.465	22	4.034	1.465	22	1.774
70	5	31	34	30.571	4	0.196	30.750	4	0.516
	10	14	16	11.832	114	1.097	11.832	51	34.19
	25	7	7	4.263	542	4.19	4.263	581	339.68
	50	4	4	2.069	125	1.795	2.069	125	2.673
	70	3	3	1.453	44	1.076	1.453	44	1.762
100	5	30	34	27.667	33	0.541	27.780	129	214.278
	10	15	17	11.122	1716	268.814	11.134	1830	4137.84
	25	7	7	4.115	2465	59.962	4.115	1241	1869.52
	50	3	4	2.042	99	10.429	2.042	99	98
	70	3	3	1.456	156	10.537	1.456	156	11.02
120	5	29	33	25.211	718	6.257	25.297	775	14.267
	10	15	16	10.650	7010	258.667	10.650	646	24.09%
	25	7	7	4.115	2561	70.925	4.115	2390	1664.14
	50	4	4	2.054	205	45.525	2.054	5	42.321
	70	3	3	1.438	170	14.175	1.438	170	14.267
150	5	27	32	23.758	1509	102.832	23.758	779	9.82%
	10	15	17	10.862	54772	4797.17	10.862	173	33.77%
	25	7	8	4.113	55590	2766.03	4.113	1771	41.58%
	50	4	5	2.014	5656	481.286	2.014	5569	299.747
	70	3	3	1.433	256	77.828	1.433	256	61.68
200	5	28	33	22.636	57703	3424.79	22.636	189	27.46%
	10	-	19	10.606	40880	32.07%	10.606	48451	42.29%
	25	-	8	4.097	61750	39.18%	4.097	44440	31.62%
	50	-	5	2.014	11751	40%	2.014	11742	40%
	70	3	3	1.434	343	392.474	1.434	343	561.37

Tabela 5.3: Comparação entre EUCUT e EUCUT$^+$ para o primeiro conjunto de instâncias de teste.

Analisando os resultados obtidos, podemos concluir que, para a variante 2-aresta conexa do PCD2CM, o algoritmo EUCUT foi o mais eficiente dentre todos os algoritmos testados. Especificamente, conseguiu resolver mais instâncias e demandou menos tempo de CPU para fazer isso. Grande parte da eficiência desse algoritmo vem do procedimento de separação implementado para o mesmo, que permitiu explorar mais nós da árvore de enumeração que os demais. No entanto, o algoritmo EUCUT$^+$ mostrou-se interessante pelo fato de produzir limitantes inferiores mais

				EUCUT			EUCUT$^+$		
$\|V\|$	dens.(%)	SO	LS	LI	NN	$t(s)$	LI	NN	$t(s)$
30	10	22	22	21.000	1	0.040	21.000	1	0.047
	25	8	8	5.000	16	0.158	6.591	1	0.039
50	5	40	41	40.000	1	0.230	40.000	3	0.307
	10	17	18	14.513	18	1.361	15.848	7	1.641
	25	8	8	5.245	11	1.061	5.245	1	0.152
70	5	32	34	31.400	7	2.539	31.400	2	23.949
	10	17	18	12.303	109	16.734	14.486	47	36.275
	25	8	9	4.907	94	72.742	7.264	10	4.889
100	5	30	36	28.465	34	20.438	28.486	51	118.060
	10	16	19	11.550	339	72.627	13.242	109	157.964
	25	8	9	5.500	94	1903.270	7.530	21	38.458
120	5	29	33	24.882	263	245.828	25.963	283	795.628
	10	17	19	11.664	3440	1718.290	13.821	919	2209.180
	25	10	11	5.555	3	45.27%	7.492	2075	6347.300
150	5	30	35	24.339	2856	1855.56	26.630	1362	6.19%
	10	-	22	12.212	1599	31.51%	14.339	846	23.72%
	25	-	10	5.623	3	38.05%	7.188	509	17.85%
200	5	-	35	23.366	662	23.02%	25.312	430	19.23%
	10	-	20	11.862	375	36.57%	13.820	95	32.69%
	25	-	10	5.646	4	43.54%	7.740	87	20%

Tabela 5.4: Comparação entre EUCUT e EUCUT$^+$ para o segundo conjunto de instâncias de teste.

fortes que aqueles obtidos por EUCUT, principalmente para o segundo conjunto de instâncias testadas.

5.2 Experimentos Para a Variante 2-Vértice Conexa

A seguir, comparamos os resultados obtidos para os algoritmos NUCUT, NUCUT$^+$ e NDCUT, implementados para resolver as formulações propostas para a variante 2-vértice conexa do PCD2CM.

Nas Tabelas 5.5 e 5.6 são apresentados os resultados obtidos pelos algoritmos NUCUT e NDCUT. A primeira tabela se refere ao primeiro conjunto de instâncias teste, enquanto a segunda diz respeito ao segundo conjunto. Essas tabelas seguem a mesma estrutura já descrita para outras tabelas desse capítulo.

Enfocando inicialmente nos resultados obtidos para o primeiro conjunto de instâncias teste, observamos que o algoritmo NUCUT foi capaz de apresentar certificados de otimalidade para todas as instâncias com até 150 vértices. Da mesma forma repetiu o desempenho para as demais instâncias com densidade 5% e aquelas com

				NUCUT			NDCUT		
$\|V\|$	dens.(%)	SO	LS	LI	NN	$t(s)$	LI	NN	$t(s)$
30	10	19	19	18.000	4	0.049	18.000	1	0.144
	25	6	7	5.253	9	0.101	5.253	9	0.310
	50	3	3	3.000	1	0.040	3.000	1	0.050
	70	3	3	1.496	13	0.183	1.496	15	0.418
50	5	15	18	14.000	9	0.139	14.000	9	0.127
	10	9	10	7.830	11	0.123	7.830	11	0.25
	25	6	6	4.390	4	0.521	6.000	1	5.522
	50	3	3	3.000	1	0.320	3.000	1	0.330
	70	3	3	1.465	22	1.668	1.465	56	3.963
70	5	31	33	30.571	13	0.498	30.571	3	0.631
	10	14	16	11.832	76	3.350	11.832	149	82.333
	25	7	7	4.263	546	7.264	4.263	564	146.200
	50	4	4	2.069	125	2.373	2.069	130	16.620
	70	3	3	1.453	44	1.406	1.453	56	4.602
100	5	30	34	27.667	223	31.814	27.667	94	191.087
	10	15	16	11.122	1855	624.719	11.122	2290	4171.53
	25	7	7	4.115	1155	23.606	4.115	1285	1858.57
	50	3	4	2.042	99	9.346	2.042	65	45.23%
	70	3	3	1.456	156	9.169	1.456	209	58.8146
120	5	29	33	25.211	818	140.784	25.225	1279	5495.23
	10	15	16	10.650	7999	468.939	10.650	2550	13.22%
	25	7	7	4.115	2582	71.042	4.115	185	36.48%
	50	4	4	2.054	205	40.887	2.054	266	5056.5
	70	3	3	1.438	170	17.622	1.438	201	73.042
150	5	27	32	23.758	520	135.479	23.758	314	15.17%
	10	-	17	10.862	13801	16.25%	10.862	420	30.89%
	25	7	7	4.113	3475	37.29%	4.113	25	44.85%
	50	4	5	2.014	5777	532.714	2.014	382	56.94%
	70	3	3	1.433	256	76.9706	1.433	283	295.051
200	5	-	31	22.636	4065	10.41%	22.636	31	26.37%
	10	-	20	10.606	13937	34.35%	10.606	92	45.18%
	25	-	8	4.097	9295	43.62%	4.097	92	52.33%
	50	4	5	2.014	13140	5298.26	2.014	17	58.54%
	70	3	3	1.434	343	808.285	1.434	52	50.91%

Tabela 5.5: Comparação entre NUCUT e NDCUT para o primeiro conjunto de instâncias

150 e 200 vértices e densidades de 50% e 70%. No total, falhou em provar a otimalidade de cinco instâncias. Já o algoritmo NDCUT falhou em 12 instâncias. Para as instâncias onde ambos os algoritmos falharam, o GAP entre os melhores limitantes obtidos durante a árvore de enumeração foi maior para o algoritmo NDCUT do que para o algoritmo NUCUT.

Para o segundo conjunto de instâncias, apenas instâncias com até 100 vértices e densidade 5% e a instância com 120 vértices e densidade de 5% tiveram a sua

				NUCUT			NDCUT		
$\|V\|$	dens.(%)	SO	LS	LI	NN	$t(s)$	LI	NN	$t(s)$
30	10	24	24	23.000	1	0.086	22.500	2	0.111
	25	9	9	5.000	13	0.199	5.000	70	1.981
50	5	41	42	39.167	2	0.158	39.600	2	0.24286
	10	19	20	14.513	21	3.821	14.789	16	5.972
	25	10	10	5.245	137	7.298	5.331	53	23.7654
70	5	33	35	32.937	2	1.586	33.545	4	10.1427
	10	18	22	12.303	77	26.140	14.228	79	101.098
	25	10	10	5.811	119	88.640	8.523	128	171.23
100	5	32	33	30.057	41	28.274	29.898	21	58.9727
	10	19	19	11.550	459	10.02%	12.172	34	26.58%
	25	11	11	5.500	132	39.94%	5.519	59	6653.5
120	5	32	34	24.883	1010	532.222	27.456	346	6.21%
	10	-	20	11.665	936	16.03%	11.915	9	31.06%
	25	-	11	5.555	3	45.27%	5.583	3	43.69%
150	5	-	36	24.340	817	13.17%	24.845	20	26.23%
	10	-	23	12.273	554	37.17%	12.297	8	46.54%
	25	-	11	5.562	3	43.68%	5.623	3	44.41%
200	5	-	36	23.367	74	33.66%	23.587	4	34.03%
	10	-	24	11.862	70	48.74%	11.866	4	49.57%
	25	-	11	5.591	36	41.76%	5.660	3	48.54%

Tabela 5.6: Comparação entre NUCUT e NDCUT para o segundo conjunto de instâncias.

otimalidade provada por NUCUT. Nenhuma instância com 150 e 200 vértices foi resolvida. No total o algoritmo NUCUT falhou em provar a otimalidade para 10 instâncias, no limite de tempo de CPU, considerado. Por outro lado, o algoritmo NDCUT apenas conseguiu provar a otimalidade de instâncias com até 100 vértices e densidade 5%, falhando um total de 10 instâncias. Nenhuma instância com 150 e 200 vértices teve a sua otimalidade provada pelo algoritmo NDCUT. Para todas as instâncias do segundo conjunto, onde ambos os algoritmos falharam, o algoritmo NDCUT obteve um GAP entre os melhores limitantes obtidos durante a árvore de enumeração maior ou igual aos do algoritmo NUCUT.

Em relação à qualidade dos limitantes inferiores obtidos, para ambos os algoritmos, uma pequena diferença foi observada entre os valores obtidos para as instâncias do primeiro conjunto. Sendo de 1,6 unidade, aproximadamente 36,82%, a maior diferença encontrada a favor do algoritmo NDCUT e, na média, sendo superior em 0,05 unidade. Para o segundo conjunto, uma pequena diferença de valores pode ser notada. No entanto, tal diferença sendo de aproximadamente 2,7 unidades, aproximadamente 31,82%, a maior diferença encontrada a favor de NDCUT e, na média, sendo superior em 0,55 unidade, aproximadamente 3,87%.

Para as instâncias onde ambos os algoritmos resolveram, o tempo de CPU médio

consumido pelo algoritmo NDCUT foi de 793,70 segundos, para o primeiro conjunto de instâncias teste e de 41,50 para o segundo. Para o algoritmo NUCUT, esses valores foram, respectivamente, de 44,71 segundos e 17,35 segundos.

Tomando a média da razão entre o número de nós explorados na árvore de enumeração e o tempo de CPU gasto por cada algoritmo para resolver cada instância, temos, para o algoritmo NUCUT, taxas de 25 e 6 nós por segundo, respectivamente para as instâncias do primeiro e segundo conjuntos. Por sua vez, para o algoritmo NDCUT, os valores obtidos são de aproximadamente 8 e 4 nós por segundo, em cada caso.

Pelos resultados computacionais obtidos, podemos concluir que o algoritmo NUCUT tem melhor desempenho que o algoritmo NDCUT.

Nas Tabelas 5.7 e 5.8 comparamos os resultados computacionais obtidos pelos algoritmos NUCUT e NUCUT$^+$. Essas tabelas seguem o mesmo padrão definido para as outras tabelas desse capítulo. Dessa forma um tempo limite de 7200 segundos de CPU é imposto ao experimento.

Os resultados obtidos para o primeiro conjunto de instâncias teste mostram que NUCUT$^+$, além de não conseguir obter certificados de otimalidade para as mesmas instâncias que NUCUT, teve o mesmo desempenho para duas instâncias adicionais para as quais NUCUT obteve êxito. No entanto, NUCUT$^+$ obteve êxito para a instância com 150 vértices e densidade 25%, ao contrário de NUCUT. Já para o segundo conjunto de instâncias, os resultados mostram que, além daquelas onde ambos os algoritmos falham, NUCUT$^+$ conseguiu obter certificados de otimalidade para as instâncias com 100 vértices e densidades 10% e 25%, ao contrário de NUCUT. Com relação aos GAPs obtidos para as instância onde ambos falham, pertencentes ao segundo conjunto de instâncias teste, NUCUT$^+$ obteve melhor desempenho.

Em relação a qualidade dos limitantes inferiores obtidos, o uso das desigualdades válidas (2.50) na formulação (2.40) não resultou em ganhos significativos nos valores das relaxações lineares para a maior parte das instâncias do primeiro conjunto de instâncias de teste. No entanto, para o segundo conjunto, ganhos foram observados, sendo a maior diferença encontrada de 2,62 unidades, aproximadamente 33,34%, e a diferença média igual à 1,43 unidade, cerca de 12,89%.

Para as instâncias onde ambos os algoritmos obtiveram êxito, o tempo computacional demandado por NUCUT$^+$ foi maior que aquele exigido por NUCUT. Este consumiu, em média, 528,80 e 182,82 segundos respectivamente para as instâncias do primeiro e do segundo conjuntos de instâncias de teste. Os valores correspondentes para NUCUT foram respectivamente de 94,10 e 68,84 segundos.

Comparando os resultados obtidos pelos algoritmos implementados para resolver a variante 2-vértice conexa do PCD2CM, notamos que os algoritmos baseado na formulação não-direcionada, NUCUT e NUCUT$^+$, foram os que se mostraram mais

				NUCUT			NUCUT$^+$		
$\|V\|$	dens.(%)	SO	LS	LI	NN	$t(s)$	LI	NN	$t(s)$
30	10	19	19	18.000	4	0.049	19.000	1	0.043
	25	6	7	5.253	9	0.101	6.000	1	0.020
	50	3	3	3.000	1	0.040	3.000	1	0.010
	70	3	3	1.496	13	0.183	1.496	13	0.094
50	5	15	18	14.000	9	0.139	14.000	9	0.113
	10	9	10	7.830	11	0.123	7.830	11	0.123
	25	6	6	4.390	4	0.521	4.391	4	0.880
	50	3	3	3.000	1	0.320	3.000	1	0.010
	70	3	3	1.465	22	1.668	3.000	22	0.249
70	5	31	33	30.571	13	0.498	30.750	4	0.250
	10	14	16	11.832	76	3.350	11.832	55	12.548
	25	7	7	4.263	546	7.264	4.263	553	20.225
	50	4	4	2.069	125	2.373	2.069	125	2.262
	70	3	3	1.4525	44	1.406	1.453	44	1.436
100	5	30	32	27.666	223	31.814	27.734	198	118.725
	10	15	17	11.122	1855	624.719	11.122	1895	4486.840
	25	7	7	4.115	1155	23.606	4.115	1128	40.318
	50	3	4	2.042	99	9.346	2.042	99	10.190
	70	3	3	1.456	156	9.169	1.456	156	9.620
120	5	29	33	25.221	818	140.784	25.221	897	1322.790
	10	15	17	10.650	7999	468.939	10.650	1983	13.92%
	25	7	7	4.115	2582	71.042	4.115	2510	111.09
	50	4	4	2.054	205	40.887	2.054	205	37.962
	70	3	3	1.438	170	17.622	1.438	170	17.791
150	5	27	32	23.758	520	135.479	23.758	959	7140.63
	10	-	17	10.862	13801	16.25%	10.862	288	32.11%
	25	7	8	4.113	3475	37.29%	4.113	3864	405.609
	50	4	5	2.014	5777	532.714	2.014	5764	502.766
	70	3	3	1.433	256	76.9706	1.433	256	91.424
200	5	-	31	22.636	4065	10.41%	22.636	250	24.77%
	10	-	20	10.606	13937	34.35%	10.606	56806	38.76%
	25	-	9	4.097	9295	43.62%	4.097	62352	29.86%
	50	4	5	2.014	13140	5298.26	2.014	22473	39.22%
	70	3	3	1.434	343	808.285	1.434	343	349.081

Tabela 5.7: Comparação entre NUCUT e NUCUT$^+$ para o primeiro conjunto de instâncias de teste.

eficientes. Notamos também que o algoritmo NUCUT$^+$ mostrou-se atraente em relação à qualidade dos limitantes duais que gera.

				NUCUT			NUCUT$^+$		
$\|V\|$	dens.(%)	SO	LS	LI	NN	$t(s)$	LI	NN	$t(s)$
30	10	24	24	23.000	1	0.086	23.000	1	0.044
	25	9	9	5.000	13	0.199	7.500	4	0.263
50	5	41	42	39.167	2	0.158	40.750	2	0.189
	10	19	20	14.513	21	3.821	16.300	6	2.250
	25	10	10	5.245	137	7.298	7.167	26	15.372
70	5	33	35	32.937	2	1.586	33.000	1	7.030
	10	18	22	12.303	77	26.140	14.626	69	89.532
	25	10	10	5.811	119	88.640	7.121	107	168.684
100	5	32	33	30.057	41	28.274	30.057	19	33.683
	10	19	19	11.550	459	10.02%	13.760	1861	2707.060
	25	11	11	5.500	132	39.94%	7.341	1568	4614.880
120	5	32	34	24.883	1010	532.222	26.615	505	1511.140
	10	-	20	11.665	936	16.03%	13.930	573	12.82%
	25	-	11	5.555	3	45.27%	7.491	133	21.95%
150	5	-	36	24.340	817	13.17%	26.963	233	11.01%
	10	-	23	12.273	554	37.17%	13.856	76	36.35%
	25	-	11	5.562	3	43.68%	5.623	118	29.06%
200	5	-	36	23.367	74	33.66%	25.551	14	33.52%
	10	-	24	11.862	70	48.74%	11.862	27	37.86%
	25	-	11	5.591	36	41.76%	7.740	79	27.27%

Tabela 5.8: Comparação entre NUCUT e NUCUT$^+$ para o segundo conjunto de instâncias de teste.

Capítulo 6

Conclusões

Dado um grafo simples não direcionado $G = (V, E)$, investigamos nesta tese o problema de encontrar conjuntos dominantes 2-{aresta, vértice} conexos de cardinalidade mínima. Tal problema é denominado Problema do Conjunto Dominante 2-Conexo Mínimo.

Dentre outras, contribuímos com sete formulações de Programação Inteira Mista para o problema. Duas destas formulações são definidas sobre o espaço das variáveis naturais do problema e se destinam a cada uma das variantes do mesmo. Três são reformulações definidas sobre grafos direcionados e duas são formulações baseadas em multi-fluxos.

Para avaliar a qualidade das diferentes formulações propostas, comparamos teoricamente a força dos limitantes produzidos por suas relaxações lineares. Dessa forma, concluímos que, para a variante 2-aresta conexa do problema, todas as formulações são equivalentes à formulação não-direcionada (2.7), com exceção da reformulação direcionada usando vértice raiz artificial (2.37).

Com o intuito de obter melhores limitantes inferiores, introduzimos desigualdades válidas para algumas das formulações propostas. Estas se baseiam nas desigualdades propostas em SIMONETTI *et al.* (2011) para o Problema do Conjunto Dominante Conexo Mínimo.

Introduzimos também duas heurísticas construtivas para a obtenção de soluções primal viáveis para o problema. Uma dessas heurísticas é uma adaptação da heurística proposta em LUCENA *et al.* (2010), para o Problema do Conjunto Dominante Conexo Mínimo. Além das heurísticas construtivas, introduzimos também um procedimento que visa reduzir a cardinalidade das soluções primais encontradas. Um experimento computacional preliminar foi conduzido para verificar qual das duas heurísticas tinha melhor desempenho. Os resultados obtidos mostraram uma pequena vantagem para a segunda heurística. No entanto, não podemos precisar de fato qual das duas heurística é a melhor, uma vez que muito provavelmente há, espaço para melhorias e ajustes finos em ambos os casos. Além disso, um experimento

computacional mais elaborado deve ser conduzido para essa avaliação.

Com o objetivo de avaliar de maneira prática a qualidade das formulações propostas, implementamos seis algoritmos *Branch and Cut*, uma para cada uma delas. Dois algoritmos foram implementados para resolver a formulação não-direcionada (2.7) e a sua reformulação direcionada usando vértice artificial (2.37), sendo ambos para a variante 2-aresta conexa do PCD2CM. Tais algoritmos foram denominados EUCUT e EDCUT, respectivamente. Uma versão do algoritmo EUCUT, chamada de EUCUT$^+$, com o reforço das desigualdades válidas (2.50) foi também implementado. Para a variante 2-vértice conexa do problema, implementamos os algoritmos NUCUT e NDCUT para resolver respectivamente a formulação não-direcionada (2.40) e a sua reformulação direcionada 2.45. Uma versão do algoritmo NUCUT com o reforço das desigualdades (2.50) foi também implementado.

Um experimento computacional foi conduzido para avaliar o comportamento dos algoritmos aqui propostos. Ao fazer isso, os algoritmos foram testados sobre dois conjuntos de instâncias construídas pseudo-aleatoriamente, introduzidas nesta tese. Analisando os resultados obtidos, concluímos que, para ambas as versões do PCD2CM, os algoritmos baseados em formulações não-direcionadas tiveram melhor desempenho.

Vale a pena ressaltar que, os algoritmos EUCUT$^+$ e NUCUT$^+$ produziram limitantes inferiores de melhor qualidade para o segundo conjunto de instâncias consideradas. Dessa maneira, tornasse válida uma investigação mais profunda sobre essas, seja efetuando um *lifting* ou tornando o procedimento de separação das mesmas mais eficiente.

Da mesma forma que acreditamos existir espaço para investigações futuras sobre desigualdades válidas adicionais para o problema, o mesmo se aplica em relação a heurísticas e procedimentos de busca local.

Por fim, uma pequena contribuição teórica foi dada com o Teorema 2.1.5 sobre as condições necessárias para que um grafo possua uma solução para a variante 2-aresta conexa do PCD2CM.

A seguir, listamos algumas investigações que poderão ser de interesse efetuar:

- Estudar e desenvolver técnicas de pré-processamento para tentar eliminar arestas e vértices que não possam fazer parte da solução ótima do problema.

- Desenvolver heurísticas primais que utilizem informação dual para encontrar limitantes superiores em cada nó da árvore de enumeração.

- Encontrar outras desigualdades válidas para as formulações propostas.

- Desenvolver algoritmos baseados no método de Benders Combinatório, introduzidos por CODATO e FISCHETTI (2006). Uma implementação preliminar mostrou-se promissora, uma vez que, conseguimos resolver todas as instâncias do primeiro conjunto de instâncias teste. No entanto, a metodologia deve ser aprimorada, já que foi incapaz de resolver qualquer instância do segundo conjunto.

- Desenvolver algoritmos de Local Branching, como os sugeridos por FISCHETTI e LODI (2003), tanto para encontrar soluções exatas para o problema, como para aprimorar as soluções primais encontradas por nossas heurísticas.

- Desenvolver metaheurísticas para encontrar soluções primais de melhor qualidade utilizando como base as heurísticas propostas nesta tese.

Referências Bibliográficas

ALZOUBI, K. M., JUN WAN, P., FRIEDER, O., 2002, "Distributed Heuristics for Connected Dominating Sets in Wireless Ad Hoc Networks", *Journal of Communications and Networks*, v. 4, pp. 22–29.

BRANDES, U., 2002, "Eager st-Ordering". In: *10th European Symposium of Algorithms (ESA'02)*, p. 247–256.

CHEN, S., LJUBIĆ, I., RAGHAVAN, S., 2010, "The regenerator location problem", *Networks*, v. 55(3), pp. 205–220.

CHERKASSKY, B. V., GOLDBERG, A. V., 1997, "On Implementing Push-Relabel Method for the Maximum Flow Problem." *Algorithmica*, v. 19, pp. 390–410.

CHIMANI, M., KANDYBA, M., LJUBIC, I., et al., 2010, "Orientation-based models for {0, 1, 2}-survivable network design: theory and practice", *Math. Program.*, v. 124, n. 1-2, pp. 413–439.

CODATO, G., FISCHETTI, M., 2006, "Combinatorial Benders' Cuts for Mixed-Integer Linear Programming", *Operations Research*, v. 54, n. 4, pp. 756–766.

COLBOURN, C. J., KEIL, J., STEWART, L. K., 1985, "Finding Minimum Dominating Cycles in Permutation Graphs", *Oper. Res. Lett.*, v. 4, n. 1, pp. 13–17.

CPLEX, 2013. "IBM ILOG CPLEX Online Documentation". Disponível em: <`http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r4/index.jsp?topic=%2Filog.odms.cplex.help%2FCPLEX%2Fmaps%2FCPLEX_1.html`>. Acessado em 10 de Junho de 2013.

CROES, G. A., 1958, "A Method fo Solving Traveling Dalesman Problems", *Operations Research*, v. 6, pp. 791–812.

DANTZIG, G., FULKERSON, R., JOHNSON, S., 1954, "Solution of a large-scale traveling-salesman problem", *Operations Research*, v. 2, pp. 393–410.

DIESTEL, R., 2010, *Graph Theory*. Springer.

DIJKSTRA, E., 1959, "A note on two problems in connexion with graphs", *Numerische Mathematik*, v. 1, n. 1, pp. 269–271.

DO FORTE, V. L., LUCENA, A., MACULAN, N., 2013, "Formulations for the Minimum 2-Connected Dominating Set Problem", *Electronic Notes in Discrete Mathematics*, v. 41, n. 0, pp. 415 – 422.

FISCHETTI, M., LODI, A., 2003, "Local branching", *Mathematical Programming*, v. 98, n. 1-3, pp. 23–47.

FORD, L. R., FULKERSON, D. R., 1956, "Maximal Flow through a Network." *Canadian Journal of Mathematics*, v. 8, pp. 399–404.

GEOFFRION, A., 2010, "Lagrangian Relaxation for Integer Programming". In: Jünger, M., Liebling, T. M., Naddef, D., et al. (Eds.), *50 Years of Integer Programming 1958-2008*, Springer Berlin Heidelberg, pp. 243–281.

GOLDBERG, A. V., TARJAN, R. E., 1988, "A New Approach to the Maximum-Flow Problem", *Journal of the Association for Computing Machinery*, v. 35, pp. 921–940.

GOMORY, R., HU, T., 1961, "Multi-Terminal Network Flows", *SIAM Journal of Applied Mathematics*, v. 9, pp. 551–570.

GOMORY, R. E., 1958, "Outline of an Algorithm for Integer Solutions to Linear Program", *Bulletin of the American Mathematical Society*, v. 64, n. 5, pp. 275–278.

GRÖTSCHEL, M., JUNGER, J., REINELT, G., 1984, "A Cutting Plane Algorithm for the Linear Ordering Problem", *Operations Research*, v. 32, pp. 1195–1220.

GRÖTSCHEL, M., MONMA, C. L., STOER, M., 1995, "Design of survivable networks". In: Ball, M. O., Monma, C. L., Nemhauser, G. (Eds.), *Handbook in Operations Research and Management Science*, v. 7, North Holland, pp. 617–671.

HARARY, F., PRINS, G., 1966, "The block-cutpoint-tree of a graph." *Publ. Math. Debrecen*, v. 13, pp. 103–107.

KOCH, T., MARTIN, A., 1998, "Solving Steiner Tree Problems in Graphs to Optimality", *Networks*, v. 32, pp. 207–232.

LAND, A., DOIG, A., 1960, "An Automatic Method for Solving Discrete Programming Problems", *Econometrica*, v. 28, pp. 497–520.

LI, Y., THAI, M. T., WANG, F., et al., 2005, "On Greedy Construction of Connected Dominating Sets in Wireless Networks: Research Articles", *Wirel. Commun. Mob. Comput.*, v. 5, n. 8, pp. 927–932.

LIANG, X., SENCUN, Z., 2007, "A Feasibility Study on Defending Against Ultra-Fast Topological Worms". In: *Proceedings of The Seventh IEEE International Conference on Peer-to-Peer Computing (P2P'07)*.

LITTLE, J. D. C., MURTY, K. G., SWEENEY, D., et al., 1963, "An Algorithm for the Traveling Salesman Problem", *Operations Research*, v. 11, pp. 972–989.

LOVÁSZ, L., 1985, "Computing ears and branchings in parallel." In: *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS'85)*, p. 464–467.

LUCENA, A., RESENDE, M., 2004, "Strong Lower Bounds for the Proize-Collecting Steiner Problem in Graphs", *Discrete Applied Mathematics*, v. 141, pp. 277–294.

LUCENA, A., MACULAN, N., SIMONETTI, L., 2010, "Reformulations and solution algorithms for the maximum leaf spanning tree problem", *Computational Management Science*, v. 7, pp. 289–311.

LUCENA, A., DA CUNHA, A. S., SIMONETTI, L., 2013, "Formulating and Solving the Minimum Dominating Cycle Problem", *Electronic Notes in Discrete Mathematics*, v. 41, pp. 423 – 430.

MACULAN, N., 1986, "A New Linear Programming Formulation for the Shortest S-Directed Spanning Tree Problem", *Journal of Combinatorics, Information and System Sciences*, v. 11, pp. 53–56.

MAGNANTI, T. L., RAGHAVAN, S., 2005, "Strong formulations for network design problems with connectivity requirements", *Networks*, v. 45, n. 2, pp. 61–79.

MILENKOVIĆ, T., MEMIŠEVIĆ, V., BONATO, A., et al., 2011, "Dominating Biological Networks", *PLoS ONE*, v. 6(8).

NEMHAUSER, G. L., WOLSEY, L. A., 1988, *Integer and combinatorial optimization*. New York, NY, USA, Wiley-Interscience. ISBN: 0-471-82819-X.

PADBERG, M., RINALDI, G., 1991, "A Branch-and-Cut Algorithm for Resolution of Large Scale of Symmetric Traveling Salesman Problem", *SIAM Review*, v. 33, pp. 60–100.

PROSKUROWSKI, ANDRZEJ, M., 1981, "Minimum dominating cycles in outerplanar graphs", *International Journal of Computer & Information Sciences*, v. 10, n. 2, pp. 127–139.

ROBBINS, H. E., 1939, "A Theorem on Graphs with an Aplication to a Problem of Traffic Control." *American Mathematical Monthly*, v. 46, pp. 281–283.

SCHMIDT, J. M., 2013, "A simple test on 2-vertex- and 2-edge-connectivity", *Information Processing Letters*, v. 113, n. 7, pp. 241 – 244.

SHANG, W., YAO, F., WAN, P., et al., 2008, "On minimum m-connected k-dominating set problem in unit disc graphs", *Journal of Combinatorial Optimization*, v. 16, n. 2, pp. 99–106.

SIMONETTI, L., CUNHA, A. S. D., LUCENA, A., 2011, "The Minimum Connected Dominating Set Problem: Formulation, Valid Inequalities and a Branch-and-Cut Algorithm". In: *INOC*, pp. 162–169.

SINHA, P., SIVAKUMAR, R., BHARGHAVAN, V., 2001, "Enhancing Ad hoc Routing with Dynamic Virtual Infrastructures". pp. 1763–1772.

SKOWROŃSKA, M., M., M., 1991, "Dominating Cycles in Halin Graphs". In: Hedetniemi, S. (Ed.), *Topics on Domination*, v. 48, *Annals of Discrete Mathematics*, Elsevier, pp. 215 – 224.

TARJAN, R. E., 1972, "Depth-[U+FB01]rst search and linear graph algorithms." *SIAM Journal on Computing*, v. 1(2), pp. 146–160.

THAI, M. T., TIWARI, R., DU, D.-Z., 2008, "On Construction of Virtual Backbone in Wireless Ad Hoc Networks with Unidirectional Links", *IEEE Transactions on Mobile Computing*, v. 7, n. 9, pp. 1098–1109.

WANG, F., THAI, M. T., DU, D.-Z., 2009, "On the construction of 2-connected virtual backbone in wireless networks", *Wireless Communications, IEEE Transactions on*, v. 8, n. 3, pp. 1230–1237.

WOLSEY, L., 1998, *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley. ISBN: 9780471283669.

WU, Y., WANG, F., THAI, M., et al., 2007, "Constructing k-Connected m-Dominating Sets in Wireless Sensor Networks". In: *Military Communications Conference, 2007. MILCOM 2007. IEEE*, pp. 1–7, Oct.