UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE COMPUTAÇÃO
PROGRAMA DE MESTRADO E DOUTORADO EM CIÊNCIAS DA COMPUTAÇÃO

LUIZ ALBERTO DO CARMO VIANA

ÁRVORE GERADORA COM DEPENDÊNCIAS MÍNIMA

FORTALEZA
2016

LUIZ ALBERTO DO CARMO VIANA

ÁRVORE GERADORA COM DEPENDÊNCIAS MÍNIMA

Dissertação de Mestrado apresentada ao Programa de Mestrado e Doutorado em Ciências da Computação, do Departamento de Computação da Universidade Federal do Ceará, como requisito parcial para obtenção do Título de Mestre em Ciências da Computação. Área de concentração: Algoritmos e Otimização.

FORTALEZA

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Ciências e Tecnologia

V667a Viana, Luiz Alberto do Carmo.
Árvore geradora com dependências mínima / Luiz Alberto do Carmo Viana. – 2016.
67 f. : il.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Departamento de Computação, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2016.
Área de Concentração: Ciência da Computação.
Orientação: Prof. Dr. Manoel Bezerra Campêlo Neto.

1. Árvores (Teoria dos grafos). 2. Programação linear. 3. Complexidade computacional. I. Título.

CDD 005

LUIZ ALBERTO DO CARMO VIANA

ÁRVORE GERADORA COM DEPENDÊNCIAS MÍNIMA

Dissertação de Mestrado apresentada ao Programa de Mestrado e Doutorado em Ciências da Computação, do Departamento de Computação da Universidade Federal do Ceará, como requisito parcial para obtenção do Título de Mestre em Ciências da Computação. Área de concentração: Algoritmos e Otimização.

Aprovada em: 31/05/2016.

BANCA EXAMINADORA

Prof. Dr. Manoel Bezerra Campêlo Neto (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Rafael Castro de Andrade
Universidade Federal do Ceará (UFC)

Prof. Dr. Tibérius de Oliveira e Bonates
Universidade Federal do Ceará (UFC)

Prof. Dr. Leonardo Sampaio Rocha
Universidade Estadual do Ceará (UECE)

Resumo

Introduzimos o problema de Árvore Geradora com Dependências Mínima, **AGDM**(G, D, w), definido sobre um grafo $G(V, E)$ e um digrafo $D(E, A)$, cujos vértices são as arestas de G e cujos arcos definem dependências entre tais arestas. O problema consiste em encontrar, dentre as árvores geradoras do grafo $G(V, E)$ que satisfaçam as restrições de dependência impostas pelo digrafo de entrada $D(E, A)$, uma que tenha custo mínimo, segundo a ponderação w das arestas de G. As restrições de dependência exigem que uma aresta e de G só pode fazer parte de uma solução se for uma fonte em D ou se fizer parte da solução *alguma* outra aresta e' tal que o arco (e', e) esteja em D. Provamos que decidir se há solução viável para **AGDM**(G, D, w) é um problema **NP-completo**, mesmo quando G é um cacto cordal e D é a união de arborescências de altura no máximo 2. Sua **NP-completude** também é mostrada ainda que G seja bipartido, as restrições de dependência ocorram apenas entre arestas adjacentes de G e formem arborescências de altura no máximo 2. Resultados idênticos são obtidos para as variantes do problema onde, nas restrições de dependência, substitui-se o requisito "alguma" por "exatamente uma" ou "toda". Para resolver o problema, apresentamos algumas formulações de programação inteira e desigualdades válidas. Propomos uma estratégia para reduzir a dimensão do problema, excluindo arestas de G com base na estrutura de D. Avaliamos os modelos e algoritmos propostos usando instâncias geradas aleatoriamente. Resultados computacionais são reportados.

Palavras-chaves: Árvore Geradora. Restrições de Dependência. Complexidade Computacional. Programação Inteira. Programação Linear.

Abstract

We introduce the Dependency Constrained Minimum Spanning Tree Problem, **DCMST**(G, D, w), defined over a graph $G(V, E)$ and a digraph $D(E, A)$, whose vertices are the edges of G and whose arcs describe dependency relations between these edges. Such problem consists of finding, among the spanning trees of $G(V, E)$ satisfying the dependency constraints imposed by $D(E, A)$, that one whose cost is minimum, according to a edge-weight function w. The dependency constraints impose that an edge e of G can be part of a solution either if it is a source in D or if *some* other edge e', such that the arc (e', e) is in D, is part of it as well. We prove that deciding whether there is a feasible solution to **DCMST**(G, D, w) is an **NP-complete** problem, even if G is a chordal cactus and D is a union of arborescences of height at most 2. **NP-completeness** also applies if G is bipartite, the dependency constraints occur only between adjacent edges of G and their related arcs describe arborescences whose height is at most 2. The same results are obtained for the problem variants which demand that, instead of "some", "exactly one"or "all"dependencies be part of a solution. To solve the problem, we introduce some integer programming formulations and some valid inequalities. We propose a strategy to reduce the problem dimension by excluding some edges of G according to the structure of D. We evaluate the introduced models and algorithms using randomly generated instances. Computational results are reported.

Key-words: Spanning Tree. Dependency Constraints. Computational Complexity. Integer Programming. Linear Programming,

Lista de Figuras

Sumário

Capítulo 1

Introdução

1.1 Motivação

O estudo de topologias de redes é motivado por problemas relativos ao planejamento de estruturas de interconexão que podem surgir em contextos bem variados, indo desde distribuição de água ou eletricidade até interligação de computadores. Em uma visão mais ampla, essas topologias buscam atender, da forma mais eficiente possível, requisitos advindos de ambientes civis e industriais, com o propósito fundamental de criar uma rede interligando entidades do universo considerado.

Neste trabalho, estudamos uma dessas topologias de redes, conhecida na literatura como árvore geradora. Tal topologia garante a existência de uma única forma de conexão entre cada par de pontos de uma dada rede. Essas são exigências que se encontram na construção de uma rede elétrica domiciliar, por exemplo, onde é desejável que todas as casas recebam eletricidade e que não haja curto-circuito em parte alguma da rede. Elas aparecem também em cenários onde se deseja uma conectividade sem redundâncias, seja por questões técnicas, de operação, manutenção, etc.

Ao partir da perspectiva abstrata para abordar a implementação real de uma Árvore Geradora, não é raro aparecer critérios que possam tornar uma árvore preferível a outra, derivados, por exemplo, da disponibilidade de orçamento e material. Reduzir o custo de construção dessa topologia de rede é a motivação fundamental de um problema clássico da literatura: encontrar a árvore geradora mínima.

Embora haja métodos de complexidade polinomial para encontrar uma árvore geradora mínima, como aqueles apresentados em [8] e [12], alguns requisitos adicionais à conectividade mínima resultam em outros problemas de maior complexidade. Exigir que certos pontos tenham um número mínimo ou máximo de ligações diretas com outros pontos são exemplos

desses requisitos, como é mostrado em [3] e [11], respectivamente.

Outro requisito capaz de tornar o problema computacionalmente difícil que pode surgir, por exemplo, devido a limitações na disponibilidade de materiais e à necessidade de que ligações entre dois pontos sejam feitas do mesmo material, é a exclusão mútua entre certas ligações, como é demonstrado em [2]. Em outras palavras, dados conjuntos de ligações conflitantes, desejamos encontrar uma Árvore Geradora em que no máximo uma ligação de cada conjunto esteja presente na árvore.

O presente trabalho propõe um novo problema relacionado, buscando desenvolver e analisar bons métodos para sua resolução. Esse novo problema surge quando acrescentamos um requisito de dependência entre as ligações que podemos utilizar para construir uma árvore geradora. Especificamente, impomos uma condição para permitir o uso de uma ligação: que seja usada pelo menos uma ligação da qual ela dependa.

1.2 Organização do texto

Esta dissertação está organizada em quatro capítulos, além dessa introdução e de uma posterior conclusão.

No Capítulo 2, introduzimos as notações e definições necessárias para a compreensão do texto. Nele abordamos conceitos de Teoria dos Grafos e Programação Matemática.

No Capítulo 3, apresentamos a definição formal do problema tratado neste trabalho, assim como provas de sua **NP-completude**. Também são exibidas propriedades do problema e casos onde ele torna-se de fácil resolução. Avaliamos sua abrangência, mostrando que outros problemas podem ser a ele reduzidos, como por exemplo o problema com restrições de conflito enunciado na seção anterior.

No Capítulo 4, são exibidas formulações para o problema de Árvore Geradora Mínima, conhecidas na literatura, que servem como base para a proposição, logo a seguir, de formulações para o problema em estudo. Também são descritas desigualdades válidas para os novos modelos.

No Capítulo 5, sumarizamos os diversos experimentos computacionais realizados ao longo do trabalho. Começamos apresentando o ambiente usado para a realização dos nossos testes, bem como as instâncias usadas. Depois mostramos os resultados obtidos com os modelos e métodos propostos no presente trabalho.

Capítulo 2

Notações e definições

Neste capítulo, definimos os conceitos e apresentamos as notações que são usadas ao longo do trabalho. Expomos aqui as bases necessárias de Teoria dos Grafos e Programação Matemática.

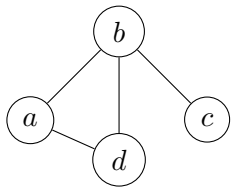
2.1 Teoria dos Grafos

Seja V um conjunto discreto finito e $V^2 = \{\{u, v\} : u, v \in V, u \neq v\}$, assim como $E \subseteq V^2$ e $E^2 = \{\{e_1, e_2\} : e_1, e_2 \in E, e_1 \neq e_2\}$. O par ordenado (V, E) define um grafo $G = (V, E)$ (usaremos a notação mais sucinta $G(V, E)$), com conjunto de vértices V e conjunto de arestas E. Definimos $n(G) = |V|$ como o seu número de vértices e $m(G) = |E|$ como o seu número de arestas. Dizemos que u e v são os extremos da aresta $\{u, v\}$.

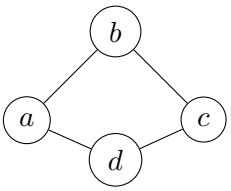
Considere agora $V' \subseteq V$ e $E' \subseteq E$. Chamamos $G'(V', E')$ um subgrafo de G, desde que $u, v \in V'$, para toda aresta $\{u, v\} \in E'$. Se $V' = V$, dizemos que G' é subgrafo gerador. Denotamos $V(E') = \{v \in V \mid \exists\, u : \{u, v\} \in E'\}$ como o conjunto de vértices que são extremos das arestas em E'. Dizemos que $G'(V(E'), E')$ é o subgrafo de G induzido pelas arestas E'. Definimos também $E(V') = \{\{u, v\} \in E : u \in V', v \in V'\}$ como o conjunto de arestas induzidas por V' (com ambos os extremos em V'). Dizemos que $G'(V', E(V'))$ é o subgrafo de G induzido por V'. Denotamos por $\delta(V') = \{\{u, v\} \in E : u \in V', v \in V \setminus V'\}$ o conjunto de arestas com exatamente um extremo em V'. Em particular, se $V' = \{v\}$, simplificamos a notação para $\delta(v)$, que descreve, pois, o conjunto de arestas incidentes a v.

Tome $G_1(V_1, E_1)$ e $G_2(V_2, E_2)$ dois grafos, com $V_1 \cap V_2 = \emptyset$. Definimos a união de G_1 e G_2 como sendo o grafo $G'(V_1 \cup V_2, E_1 \cup E_2)$.

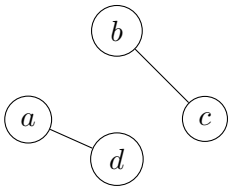
Seja v um vértice de G. Definimos a vizinhança de v em G como $N(v) = \{u \in V : \{u, v\} \in E\}$. Definimos o grau de v como $d(v) = |N(v)|$. Se $d(v) = 1$, dizemos que v é



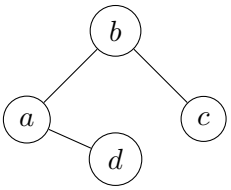
(a) grafo conexo.



(c) grafo cíclico.



(b) grafo desconexo.



(d) grafo acíclico.

Figura 2.1: Ilustrações de grafos conexos e grafos acíclicos.

uma folha em G. Denotamos $N_0(v) = \{v\}$, $N_1(v) = \{v\} \cup N(v)$ e $N_k(v) = \bigcup_{u \in N_{k-1}(v)} N(u)$, para $k \geq 2$. Assim, $N_k(v)$ são os vértices que estão a uma distância máxima k de v, sendo $d(u, v) = \min\{k : u \in N_k(v)\}$ a distância entre vértices u e v. Note que $d(u, v) = d(v, u)$ e $d(u, v) = \infty$ se $u \notin N_k(v)$ para todo $k \in \mathbb{N}$.

Chamamos o grafo G de conexo se, para todos $u, v \in V$, $d(u, v)$ é finito; caso contrário, G é dito desconexo. As figuras 2.1(a) e 2.1(b) são exemplos de grafo conexo e desconexo, respectivamente. Definimos como componentes de um grafo G seus subgrafos conexos maximais. Um subconjunto de arestas $E' \subseteq E$ de um grafo $G(V, E)$ é dito um corte se o subgrafo $G'(V, E \setminus E')$ tem mais componentes que G. Se E' é um corte composto por uma única aresta e, então e é dita uma aresta de corte de G.

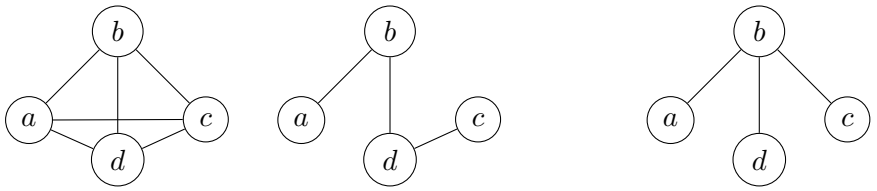
Um grafo P é dito um caminho se P é conexo, o grau de exatamente dois de seus vértices é 1, e o grau de seus outros vértices é 2. O comprimento de P é o seu número de arestas ou, equivalentemente, seu número de vértices menos um. Se G é conexo, então, para quaisquer vértices u e v, há um subgrafo de G que é um caminho P de tamanho $d(u, v)$, onde u e v são exatamente os vértices de grau 1 de P. Dizemos que P é um caminho entre u e v.

Definimos a excentricidade de um vértice $v \in V$ como $\epsilon(v) = \max\{d(u, v) : u \in V \setminus \{v\}\}$. Definimos o raio de um grafo conexo G como a menor de suas excentricidades e, analogamente, definimos seu diâmetro como a maior de suas excentricidades.

Um grafo G é dito acíclico se existe no máximo um caminho simples entre qualquer par de vértices $u, v \in V$. Grafos acíclicos também são conhecidos como florestas. A Figura 2.1(d) traz um exemplo de grafo acíclico. Caso não seja acíclico, o grafo é dito cíclico. A Figura 2.1(c) é um exemplo de um grafo cíclico.

Um grafo é dito um ciclo se é conexo e todo vértice tem grau 2. O comprimento de um ciclo é o seu número de arestas, que é igual ao seu número de vértices. Um ciclo pode ser visto, portanto, como dois caminhos simples entre um mesmo par de vértices u e v. Caso $d(u, v) = k$ e a remoção de um caminho simples de comprimento k entre u e v implique em $d(u, v) = l$, então existe um ciclo de comprimento $k + l$ em G. Note que G é acíclico se, e somente se, não possui qualquer ciclo. Se não existe um ciclo de comprimento $k \geq 4$ em G, então G é dito cordal. Assim, um grafo cordal é acíclico ou todos os seus ciclos são triângulos (ciclos de tamanho 3). O grafo cacto é aquele em que quaisquer dois de seus ciclos não tem aresta em comum.

Chamamos G de árvore se G é conexo e acíclico, ou seja, se existe um e somente um caminho simples entre qualquer par de vértices $u, v \in V$. Uma árvore geradora de G é um subgrafo gerador que é uma árvore. As figuras 2.2(b) e 2.2(c) ilustram árvores geradoras do grafo apresentado na Figura 2.2(a). Dizemos que um vetor $x \in \mathbb{B}^{m(G)}$ é de incidência de uma árvore geradora $T(V, E')$ de $G(V, E)$ se $x_e = 1 \iff e \in E', \forall e \in E$.



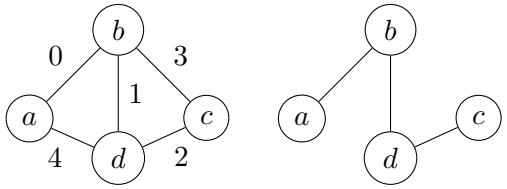
(a) Grafo G. (b) Uma árvore geradora de G. (c) Outra árvore geradora de G

Figura 2.2: Ilustração de árvore geradora.

Um grafo T é dito *caterpillar* se T é uma árvore e a remoção de suas folhas resulta em um grafo caminho. Equivalentemente, T é *caterpillar* se todo vértice de grau pelo menos 3 tem no máximo dois vizinhos que não são folhas.

Seja $w : E \to \mathbb{R}$ uma função associando um número real a cada aresta de G. Para $e \in E$, definimos $w(e)$ como o peso da aresta e. Usaremos alternativamente a notação w_e para representar $w(e)$. Denotamos por $w(G) = \sum_{e \in E} w_e$ o peso do grafo G. Uma árvore geradora T de G é dita mínima se $w(T) \leq w(T')$, para toda árvore geradora T' de G. O problema de encontrar a árvore geradora mínima de um grafo G, considerando a função de peso w sobre suas arestas, é abreviado como $\mathbf{AGM}(G, w)$.

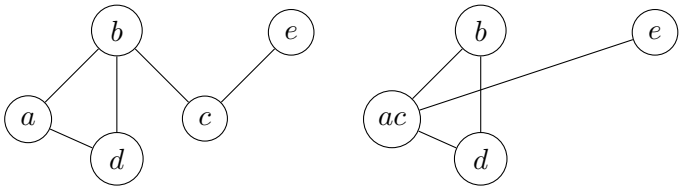
Dados um grafo $G(V, E)$ e vértices $u, v \in V$, definimos o grafo $G'(V', E')$ obtido pela contração de u e v como $V' = V \cup \{uv\} \setminus \{u, v\}$ e $E' = E \setminus (\delta(u) \cup \delta(v)) \cup \{\{a, uv\} : a \in N(u) \cup N(v)\}$. Intuitivamente, G' é um grafo que se obtém trocando-se u e v por um vértice



(a) Grafo G. (b) Árvore geradora mínima de G.

Figura 2.3: Ilustração de árvore geradora mínima.

uv, que terá como vizinhança a união das vizinhanças de u e v. Assim, podemos definir também o grafo G' obtido pela contração da aresta $e = \{u, v\}$ de G como aquele obtido pela contração dos vértices u e v. Ver figuras 2.4(a) e 2.4(b). Dada uma aresta $e = \{u, v\} \in E$, podemos definir também o grafo $G'(V', E')$ obtido pela subdivisão de e como $V' = V \cup \{v_e\}$ e $E' = E \cup \{\{u, v_e\}, \{v, v_e\}\} \setminus \{\{u, v\}\}$.



(a) Grafo G. (b) Grafo G após contração de a e c.

Figura 2.4: Ilustração da operação de contração em um grafo.

Seja $G(V, E)$ um grafo. Definimos o grafo linha de G como $G_L(E, E')$, onde $E' = \{\{e_1, e_2\} \in E^2 : e_1 \cap e_2 \neq \emptyset\}$. Note que a definição de E' implica que existe aresta entre e_1 e e_2 em G_L se, e somente se, e_1 e e_2 têm exatamente uma extremidade em comum.

Tomando V um conjunto discreto, seja $A \subseteq \{(u, v) \in V \times V, u \neq v\}$. Chamamos $D(V, A)$ o digrafo formado pelos vértices ou nós V e pelos arcos A. Denotamos por $N^+(v) = \{u : (v, u) \in A\}$ o conjunto de vértices diretamente atingíveis a partir de v. Da mesma forma, denotamos por $N^-(v) = \{u : (u, v) \in A\}$ o conjunto de vértices a partir dos quais pode-se atingir diretamente o vértice v. Chamamos v de fonte se $N^-(v) = \emptyset$, e de sumidouro se $N^+(v) = \emptyset$. Definimos também $N_0^+(v) = \{v\}, N_1^+(v) = \{v\} \cup N^+(v)$ e, para $k \geq 2, N_k^+(v) = \bigcup_{u \in N_{k-1}^+(v)} N^+(u)$. Denotamos $d(u, v) = \min\{k : v \in N^+(u)\}$ como a distância direcionada do vértice u para o vértice v. De forma análoga, definimos $N_0^-(v), N_1^-(v)$ e $N_k^-(v), k \geq 2$. Podemos definir também a excentricidade de um vértice v como a maior distância direcionada de v a cada outro vértice u.

Considera agora $V' \subseteq V$ e $A' \subseteq A$. Chamamos $D'(V', A')$ um subdigrafo de D, desde

que $u, v \in V'$, para todo arco $(u, v) \in A'$. Se $V' = V$, dizemos que D' é subdigrafo gerador. Denotamos $V(A') = \{u \in V \mid \exists v : \{(u, v), (v, u)\} \cap A' \neq \emptyset\}$ como o conjunto de vértices que são origem ou destino dos arcos em A'. Dizemos que $D'(V(A'), A')$ é o subdigrafo de D induzido pelos arcos A'. Definimos também $A(V') = \{(u, v) \in A : u, v \in V'\}$ como os conjuntos de arcos que saem e chegam em vértices de V'. Dizemos que $D'(V', A(V'))$ é o subdigrafo de D induzido por V'. Definimos também $\delta^+(v) = \{(v, b) \in A : b \in V\}$ e $\delta^-(v) = \{(b, v) \in A : b \in V\}$ como os conjuntos de arcos que, respectivamente, saem de e chegam a v. Da mesma forma, para $S \subseteq V$, definimos $\delta^+(S) = \{(u, v) : u \in S, v \in V \setminus S\}$ e $\delta^-(S) = \{(u, v) : v \in S, u \in V \setminus S\}$ como os conjuntos de arcos que, respectivamente, saem de e chegam a algum vértice de S.

Tome $D_1(V_1, A_1)$ e $D_2(V_2, A_2)$ dois digrafos, com $V_1 \cap V_2 = \emptyset$. Definimos a união de D_1 e D_2 como sendo o digrafo $D'(V_1 \cup V_2, A_1 \cup A_2)$.

Para um vértice v, definimos seu grau de saída como $d^+(v) = |N^+(v)|$. Similarmente, seu grau de entrada é $d^-(v) = |N^-(v)|$.

Seja $G(V, E)$ um grafo qualquer. Para A tal que $\{u, v\} \in E$ implica ou $(u, v) \in A$ ou $(v, u) \in A$, chamamos $D(V, A)$ de uma orientação de G. Ver Figura 2.5. De forma similar, um grafo $G(V, E)$ é dito subjacente a um digrafo $D(V, A)$ se $E = \{\{u, v\} : (u, v) \in A\}$.

(a) Um grafo G. (b) Uma orientação de G.

Figura 2.5: Exemplo da orientação de um grafo.

Um digrafo $D(V, A)$ é dito uma arborescência se D é orientação de uma árvore T tal que $N^+(u) \cap N^+(v) = \emptyset$, para $u, v \in V$. Note que uma arborescência tem exatamente uma fonte, denotada como sua raiz. O grafo da Figura 2.5(b) sem o arco (b, d) é uma arborescência. Uma arborescência $D(V, A)$ é uma estrela centrada no vértice v se $d^+(v) > 0$ e $d^+(u) = 0$, para todo $u \in V, u \neq v$. A Figura 2.6 ilustra esse conceito. A altura de uma arborescência é definida como a maior distância de sua fonte até uma de suas folhas.

Um digrafo D é um ciclo orientado se D é a orientação de um ciclo tal que $d^+(v) = d^-(v)$, para todo $v \in V$. D é dito um caminho orientado se pode ser obtido a partir de um ciclo orientado pela remoção de um único vértice ou de um único arco.

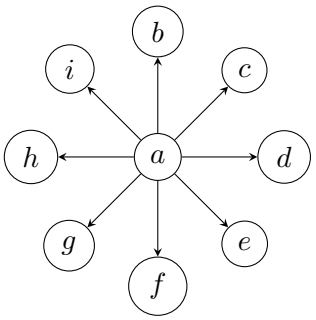


Figura 2.6: Exemplo de arborescência estrela.

2.2 Programação Matemática

Sejam $S \subseteq \mathbb{R}^n$ um conjunto e f uma função. Definimos o problema de minimização de f sobre S como determinar $z = \min\{f(x) : x \in S\}$, ou seja, encontrar o menor valor que f pode assumir em S e, usualmente, determinar também um ponto de S onde f assume esse valor.

Nesse contexto, f é chamada de função objetivo do problema. O conjunto S é chamado conjunto de soluções viáveis. Cada ponto em S é uma solução viável e aquelas que minimizam f são soluções ótimas.

Consideramos todos os vetores nesse trabalho como vetores-coluna. Sejam $a \in \mathbb{R}^n, a \neq 0$, $b \in \mathbb{R}$. A equação da forma $a^T x = b$ é dita linear, e o conjunto de vetores x que a satisfazem é dito hiperplano. Similarmente, a inequação da forma $a^T x \leq b$ é dita linear, e o conjunto de vetores x que a satisfazem é chamado semiespaço.

Um conjunto $P \subset \mathbb{R}^n$ é dito um poliedro se pode ser definido pela interseção de um número finito de semiespaços, ou seja, P é da forma $\{x \in \mathbb{R}^n : a_i^T x \leq b_i, 1 \leq i \leq m\}$, para $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, $1 \leq i \leq m$, para algum $m \in \mathbb{N}$. Uma função $f : \mathbb{R}^n \to \mathbb{R}$ é dita linear se ela pode ser representada na forma $f(x) = c^T x$, para algum $c \in \mathbb{R}^n$.

Dada uma função linear $f(x) = c^T x$ e um poliedro $P \subseteq \mathbb{R}^n$, definimos um problema linear de minimização como sendo da forma $z = \min\{f(x) : x \in P\}$. Nesse contexto, as equações lineares $a_i^T x \leq b_i$ que descrevem P são também chamadas restrições. Usualmente, considera-se que P está contido no primeiro octante, ou seja, que $x \geq 0$ está entre as restrições de P (se necessário pode-se fazer uma mudança de variáveis para obter essa condição).

Seja $P \subseteq \mathbb{R}^n_+$ um poliedro da forma $\{x \in \mathbb{R}^n_+ : a_i^T x \leq b_i, 1 \leq i \leq m\}$, $m \in \mathbb{N}$. Seja também $A \in \mathbb{R}^{m \times n}$ a matriz cuja i-ésima linha é o vetor a_i^T, e $b \in \mathbb{R}^m$ o vetor cuja i-ésima entrada é b_i, $1 \leq i \leq m$. O problema linear de minimização com restrições apresentado no

parágrafo anterior pode também ser representado da forma abaixo.

$$\min \quad f(x) = c^T x \tag{2.1}$$

$$\text{s.a:} \quad Ax \leq b \tag{2.2}$$

$$x \in \mathbb{R}^n_+ \tag{2.3}$$

Nessa representação, A é chamada a matriz de coeficientes das restrições, e b o vetor de termos independentes das restrições. c é denominado o vetor de coeficientes da função objetivo ou vetor de custos.

Dada uma função linear $f(x) = c^T x$ e $P \subseteq \mathbb{R}^n$ um poliedro da forma $\{x \in \mathbb{R}^n : a_i^T x \leq b_i, 1 \leq i \leq m\}$, $m \in \mathbb{N}$, definimos um problema linear inteiro de minimização como sendo da forma $z = \min\{f(x) : x \in P, x \in \mathbb{Z}^n_+\}$. Sendo A e b definidos como acima, tal problema linear inteiro com restrições pode ser representado da forma abaixo.

$$\min \quad f(x) = c^T x \tag{2.4}$$

$$\text{s.a:} \quad Ax \leq b \tag{2.5}$$

$$x \in \mathbb{Z}^n_+ \tag{2.6}$$

Podemos notar que o espaço viável do problema linear inteiro acima é $P \cap \mathbb{Z}^n_+$. Sejam $P_1 \subset P$ e $P_2 \supset P$ poliedros tais que $P_1 \cap \mathbb{Z}^n_+ = P \cap \mathbb{Z}^n_+ = P_2 \cap \mathbb{Z}^n_+$. Dizemos que P_1 e P_2 dão origem a formulações equivalentes à de P, mais forte e mais fraca, respectivamente.

Dado o problema linear inteiro de minimização definido por uma função linear $f(x) = c^T x$ e um poliedro $P \subseteq \mathbb{R}^n_+$, dizemos que sua relaxação linear é o problema linear de minimização definido por f e P. Seja $a^T x \leq b_0, a \in \mathbb{R}^n, b_0 \in \mathbb{R}$ uma inequação linear cujo semiespaço é $H \subseteq \mathbb{R}^n$. Dizemos que $a^T x \leq b_0$ é uma desigualdade válida para o problema linear inteiro de minimização definido por f e P se $P \cap \mathbb{Z}^n_+ = (P \cap H) \cap \mathbb{Z}^n_+$. Note que $P \cap H$ também é um poliedro, logo dá origem a uma formulação equivalente, potencialmente mais forte, já que pode ocorrer de $P \cap H$ ser estritamente contido em P.

Capítulo 3

O problema

Neste capítulo, introduzimos o problema de Árvore Geradora com Dependências. Apresentamos a definição formal do problema, ilustrada com exemplos simples e motivada por uma aplicação prática. Exibimos ainda uma análise de sua complexidade computacional e alguns casos particulares de fácil resolução. Discutimos, particularmente, estratégias para encontrar uma solução viável, problema que se mostra difícil. Finalmente, consideramos variações e problemas relacionados, procurando evidenciar a abrangência do problema aqui definido.

3.1 Definição

Nesta seção, definimos dois problemas que serão explorados durante esta dissertação: **AGDM**(G, D, w), e sua versão de decisão, **AGD**(G, D). Eles se referem ao conceito de árvore geradora com dependências, que introduzimos a seguir.

Definição 3.1. *Seja $G(V, E)$ um grafo e seja $D(E, A)$ um digrafo cujos vértices são as arestas de G. Dizemos que D é um digrafo de dependências (das arestas de G). Dizemos também que $e_2 \in E$ é uma dependência de $e_1 \in E$, se o arco (e_2, e_1) está em D (a ocorrência do arco (e_2, e_1) indica que e_1 depende de e_2). Uma árvore geradora de G com dependências D é uma árvore geradora $T(V, E')$ de G, tal que $e_1 \in E'$ implica que e_1 é uma fonte em D ou existe $e_2 \in E'$ tal que $(e_2, e_1) \in A$. Em outras palavras, uma aresta pode fazer parte de T apenas se ela não tem dependências ou, caso tenha, se ao menos uma delas também faz parte de T.*

Problema 3.1. *Seja $G(V, E)$ um grafo e $w : E \to \mathbb{R}^+$ uma função de peso para suas arestas. Seja $D(E, A)$ um digrafo cujos vértices são as arestas de G. O problema da Árvore Geradora*

com Dependências Mínima, abreviado como $\boldsymbol{AGDM}(G, D, w)$*, consiste em encontrar uma árvore de custo mínimo entre as árvores geradoras de* G *com dependências* D*.*

Problema 3.2. *Seja* $G(V, E)$ *um grafo e* $D(E, A)$ *um digrafo cujos vértices são as arestas de* G*. O problema da Árvore Geradora com Dependências, abreviado como* $\boldsymbol{AGD}(G, D)$*, consiste em decidir se existe uma árvore geradora de* G *com dependências* D*.*

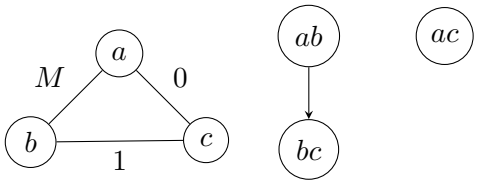
3.2 Exemplos

Nesta seção, apresentamos dois exemplos para $\mathbf{AGDM}(G, D, w)$. O primeiro exemplo é viável, enquanto o segundo é inviável.

3.2.1 Exemplo viável

Considere a instância de $\mathbf{AGDM}(G, D, w)$ com G dado pela Figura 3.1(a) e D dado pela Figura 3.1(b). Temos que G tem três árvores geradoras, dadas pela Figura 3.2.

A Figura 3.2(a) representa a árvore geradora mínima de G, que é inviável para $\mathbf{AGDM}(G, D, w)$, pois a aresta $\{b, c\}$ está presente sem sua dependência $\{a, b\}$. Já na Figura 3.2(b) está representada uma árvore geradora de G que é viável para $\mathbf{AGDM}(G, D, w)$, pois a aresta $\{a, b\}$ não tem dependências a cumprir e é dependência de $\{b, c\}$. Por fim, a Figura 3.2(c) representa uma árvore geradora de G viável para $\mathbf{AGDM}(G, D, w)$, pois suas arestas $\{a, b\}$ e $\{a, c\}$ não têm dependências a cumprir. Em particular, note que essa última é a solução ótima do problema, com custo M, enquanto a árvore geradora mínima de G tem custo 1. Isso mostra que a diferença entre os valores ótimos da árvore geradora e da árvore geradora com dependências pode ser arbitrariamente grande.

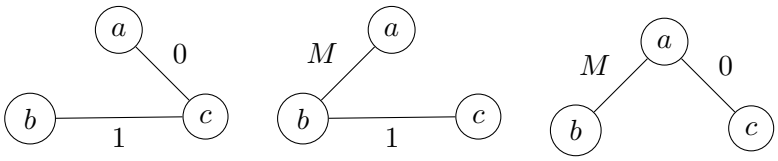


(a) $G(V, E)$ (b) $D(E, A)$

Figura 3.1: Instância viável de $\mathbf{AGDM}(G, D, w)$.

3.2.2 Exemplo inviável

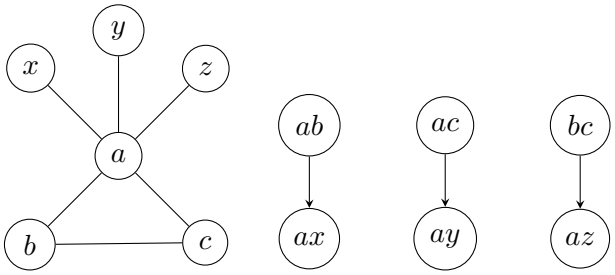
Seja $\mathbf{AGDM}(G, D, w)$ com G dado pela Figura 3.3(a) e D dado pela Figura 3.3(b). Notamos que essa instância é inviável.



(a) Solução inviável. (b) Solução viável. (c) Solução ótima.

Figura 3.2: Soluções para $\mathbf{AGDM}(G, D, w)$: (a) é árvore geradora mínima de G, no entanto é inviável; (b) e (c) são árvores geradoras de G, e ambas satisfazem as restrições de dependência impostas por D.

Observe que as arestas $\{a, x\}$, $\{a, y\}$ e $\{a, z\}$ são arestas de corte em G. Desse modo, toda árvore geradora de G deve conter essas arestas. Suas dependências, respectivamente $\{a, b\}$, $\{a, c\}$ e $\{b, c\}$, formam um ciclo em G, e portanto toda árvore geradora de G não contém todas elas, implicando que toda árvore geradora de G viola alguma restrição de dependência, seja de $\{a, x\}$, $\{a, y\}$ ou $\{a, z\}$. Desse modo, temos que $\mathbf{AGDM}(G, D, w)$ não tem soluções viáveis, sendo portanto inviável.



(a) $G(V, E)$. (b) $D(E, A)$.

Figura 3.3: Instância inviável de $\mathbf{AGDM}(G, D, w)$.

3.3 Aplicação

Antes de mais nada, vamos justificar a definição e o estudo de $\mathbf{AGDM}(G, D, w)$ com alguma aplicação prática. Nesta seção, descrevemos um cenário onde é possível aplicá-lo.

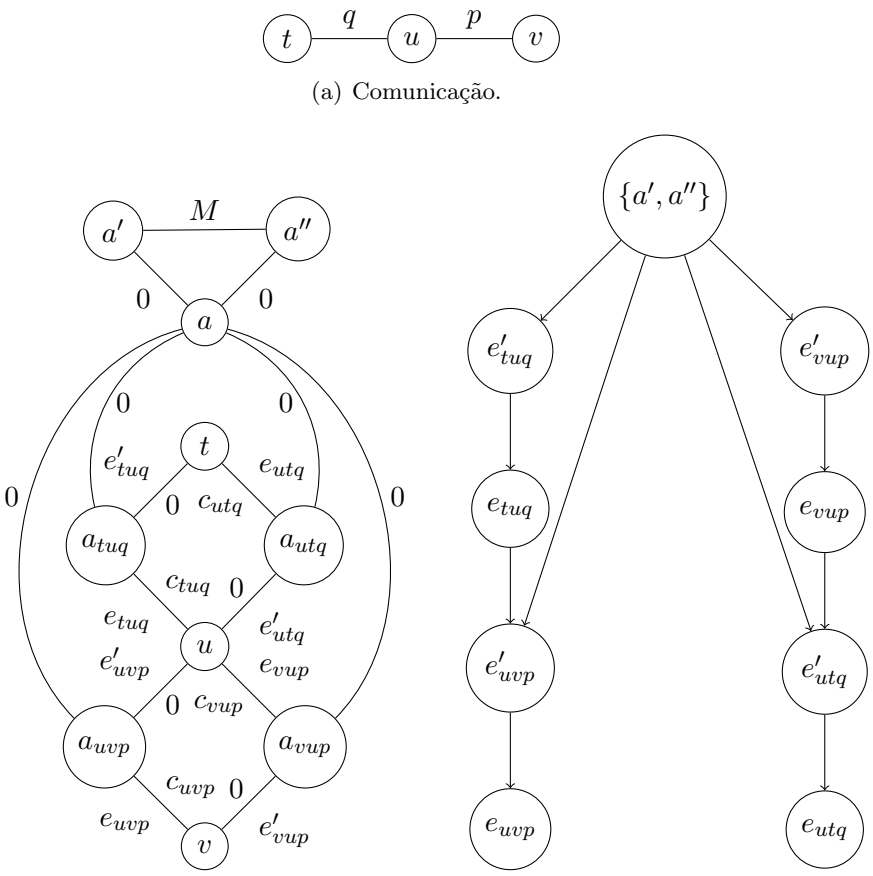
Suponha que existe uma rede de dispositivos que podem se comunicar entre si usando diferentes protocolos de comunicação. Cada dispositivo u consegue ler um conjunto de protocolos P_u e é capaz de realizar conversões entre alguns desses protocolos. Uma mensagem só é aceita por um dispositivo u, se a ele chega através de um protocolo p em P_u; e só poderá ser retransmitida para um dispositivo v através de um protocolo q, simultaneamente em P_u e P_v, tal que u possa converter entre p e q ou $p = q$. Entre cada par de dispositivos u e v há um custo de comunicação c_{uvp} que depende do protocolo p a ser usado - cada protocolo pode

ter seu próprio custo de codificação e decodificação, por exemplo. O problema consiste em, a partir de um dispositivo s, fazer chegar a todos os demais uma certa mensagem, segundo uma topologia em árvore de custo mínimo, de modo que cada dispositivo receba a mensagem em um protocolo por ele conhecido e a retransmita, se necessário, através de um protocolo para o qual saiba converter. Supomos sempre possível "conversão"de um protocolo para ele mesmo.

Formalmente, a entrada do problema pode ser descrita por um grafo H, cujos vértices representam os dispositivos e cujas arestas representam as conexões entre eles. A cada vértice u é associado um conjunto P_u de protocolos conhecidos por u e um subconjunto T_u de pares ordenados de P_u, que representa as conversões realizáveis por u. Uma solução viável para o problema é um conjunto S de tuplas (u, v, p), indicando que u envia a mensagem a v pelo protocolo p, tal que: (i) $\{(u, v) : (u, v, p) \in S\}$ induz uma arborescência cujo grafo subjacente é uma árvore geradora de H; (ii) se $(u, v, p) \in S$ então $p \in P_u \cap P_v$ e, se $u \neq s$, existe $(w, u, q) \in S$ com $(q, p) \in T_u$. Uma solução ótima para esse problema é um conjunto de tuplas que é uma solução viável e que minimiza a soma dos custos c_{uvp}, para cada tupla (u, v, p).

Podemos modelar esse problema como uma instância $\mathbf{AGDM}(G, D, w)$. Construímos $G(V, E)$ da seguinte forma: criamos três vértices artificiais a, a' e a'' - para induzir as arestas que indicarão a viabilidade da solução - e um vértice u para cada dispositivo u. Seja s o dispositivo de onde partirá uma mensagem. Se existe a possibilidade de comunicação entre u e v pelo protocolo p, criamos vértices a_{uvp} e a_{vup}, além de arestas $e'_{uvp} = \{u, a_{uvp}\}, e_{uvp} = \{a_{uvp}, v\}, e'_{vup} = \{v, a_{vup}\}, e_{vup} = \{a_{vup}, u\}$ com pesos $w(e'_{uvp}) = w(e'_{vup}) = 0, w(e_{uvp}) = w(e_{vup}) = c_{uvp}$; criamos ainda arestas entre a e todo vértice a_{uvp}, além de $\{a, a'\}, \{a, a''\}$ e $\{a', a''\}$, todas essas com peso 0, à exceção de $\{a', a''\}$, que tem peso arbitrariamente grande M. Ver Figura 3.4(b). A ideia é que a escolha de e_{uvp} para uma solução de $\mathbf{AGDM}(G, D, w)$ indique, no problema original, que a mensagem foi enviada de u para v segundo o protocolo p. Similarmente, a escolha de e_{vup} vai significar que a mensagem foi enviada de v para u segundo p. Para isso, construímos $D(E, A)$ da seguinte forma: para todo par de arestas e_{uvp} e e'_{uvp}, criamos um arco (e'_{uvp}, e_{uvp}) e, se $u \neq s$, também o arco $(\{a', a''\}, e'_{uvp})$, que dizemos ser uma dependência artificial; se o dispositivo $u, u \neq s$, é capaz de converter entre os protocolos p e q e pode se comunicar com v e t pelos protocolos p e q, respectivamente, criamos arcos (e_{tuq}, e'_{uvp}) e (e_{vup}, e'_{utq}). Ilustramos a construção de D na Figura 3.4(c).

Proposição 3.1. *Toda solução viável do problema descrito corresponde a uma solução viável*



(a) Comunicação.

(b) Grafo G.

(c) Digrafo D, $s \notin \{t, u, v\}$.

Figura 3.4: Ilustração da modelagem do problema de conversão de protocolos em um **AGDM**(G, D, w). Se $t = s$, não há o arco $(\{a', a''\}, e'_{tuq})$ em D. A origem da mensagem é s.

de $\boldsymbol{AGDM}(G, D, w)$ *de igual custo. O mesmo acontece no sentido inverso caso a solução de* $\boldsymbol{AGDM}(G, D, w)$ *não use a aresta* $\{a', a''\}$.

Demonstração. Tome uma solução viável para o problema descrito. Podemos obter uma solução viável para $\mathbf{AGDM}(G, D, w)$ da seguinte forma: para cada tupla (u, v, p) da solução - indicando que a mensagem é passada do dispositivo u para o dispositivo v por meio do protocolo p - selecionamos as arestas e'_{uvp} e e_{uvp}. Como a mensagem chega em u, $u \neq s$, por um protocolo a partir do qual u poderia converter para p, temos que a dependência de todas as arestas selecionadas está satisfeita (chamamos a árvore formada de T); para cada vértice da forma a_{uvp} que ficou isolado, selecionamos a aresta $\{a, a_{uvp}\}$, além das arestas $\{a, a'\}$ e $\{a, a''\}$, criando uma estrela centrada em a. Unimos essa estrela a T selecionando uma aresta entre a e um vértice a_{uvp} de T. O grafo assim formado é uma árvore geradora de G que satisfaz as dependências descritas por D. Ver Figura 3.5. Como as únicas arestas com custo não nulo são as e_{uvp}, e como foi selecionada exatamente uma para cada transmissão correspondente da mensagem, temos que o custo dessa solução viável de $\mathbf{AGDM}(G, D, w)$ é o mesmo da solução viável do problema descrito.

Tome agora uma árvore T viável para $\mathbf{AGDM}(G, D, w)$ que não use $\{a', a''\}$. Suponha que a aresta e_{uvp} está em T. Então, pelas restrições de dependência, também está e'_{uvp} e, por conseguinte, sendo $u \neq s$, T contém ainda alguma aresta e_{tuq}, relativa a um dispositivo t que pode transmitir para u via protocolo q, que pode ser convertido em p por u. Além disso, e_{vup} não pertence a T, pois se pertencesse, e'_{vup} também pertenceria, pelas dependências, formando então um ciclo. Assim, podemos mapear a escolha de e_{uvp} na transmissão da mensagem de u para v via p, e com isso, obter uma solução viável para o problema. A solução é obtida removendo todos os vértices a_{uvp} que são folhas, além de a, a' e a'', o que resulta em uma árvore, e depois contraindo cada aresta e'_{uvp} remanescente. Todas as arestas eliminadas ou contraídas têm custo zero, mantendo assim o custo original de T. □

3.4 Complexidade

Nesta seção, estudamos a complexidade de $\mathbf{AGD}(G, D)$. Mostramos **NP-completude** mesmo para classes de grafos bem restritas. Evidentemente, o problema é bem simples se G é uma árvore: basta verificar se G satisfaz as dependências D, o que pode claramente ser feito em tempo polinomial. Consideramos então o problema em superclasses de árvores - classes de grafos em que as árvores estão contidas - como bipartidos e cactos. Além disso, procuramos

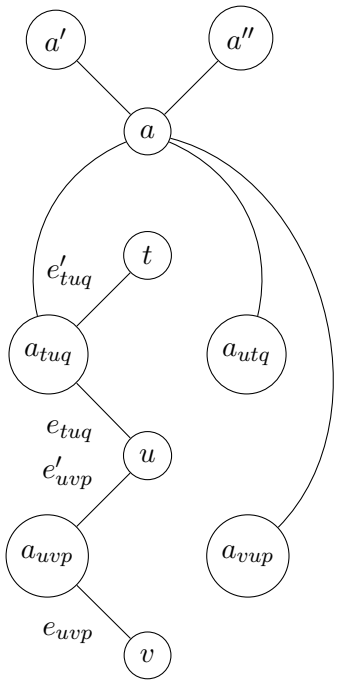


Figura 3.5: Ilustração de como seria a comunicação $t \to u \to v$, usando o protocolo q e em seguida o protocolo p, em uma solução viável para o problema de comunicação.

manter D também restrito, por um lado, a digrafos simples, como união de arborescências, por outro a digrafos possivelmente mais afeitos às aplicações, onde as dependências ocorrem entre arestas adjacentes.

3.4.1 Redução para cactos cordais

A **NP-completude** de **AGD**(G, D) implica diretamente que **AGDM**(G, D, w) é **NP-difícil**, já que este é a versão de otimização daquele problema de decisão. Em outras palavras, determinar a solução de menor peso para **AGDM**(G, D, w) é pelo menos tão difícil quanto encontrar uma solução qualquer para **AGD**(G, D).

Aqui trabalhamos com **3-SAT 1 em 3**, que é um problema **NP-completo**, como é mostrado em [6]. Ele consiste em satisfazer uma entrada para o problema **3-SAT** de tal forma que cada uma de suas cláusulas tenha exatamente um literal verdadeiro. Definimos analogamente **3-SAT 2 em 3**, onde a entrada deve ser satisfeita tal que toda cláusula deve ter exatamente dois literais verdadeiros.

Podemos reduzir uma entrada de **3-SAT 1 em 3** para uma de **3-SAT 2 em 3** aplicando negação a todos os literais em todas as cláusulas da entrada. De fato, se a instância de **3-SAT 1 em 3** é satisfazível, a mesma valoração que a satisfaz também satisfaz a instância associada de **3-SAT 2 em 3**, pois há exatamente dois literais valorados como falsos em cada

cláusula, e esses passam a ser valorados como verdadeiros após a transformação. Da mesma forma, se a entrada de **3-SAT 1 em 3** não é satisfazível, toda valoração faz com que uma cláusula tenha 0, 2 ou 3 literais verdadeiros, que passa a ter 3, 1 ou 0 literais verdadeiros, respectivamente, quando aplicamos negação aos seus literais. Assim, temos que **3-SAT 2 em 3** é **NP-completo**.

Antes de prosseguir, notemos que uma mesma instância de **3-SAT 2 em 3**, uma fórmula, pode ser escrita de diversas maneiras, obtidas por permutações dos literais de suas cláusulas. Vamos escolher uma forma específica para representar cada cláusula em uma instância de **3-SAT 2 em 3**. Dado que a instância tem n variáveis, tome uma ordenação $x_1, x_2, \ldots, x_n$ de suas variáveis. Gostaríamos de considerar, a partir de agora, a representação das instâncias de **3-SAT 2 em 3** em que os literais de cada cláusula ocorrem em ordem crescente, em relação à ordenação estabelecida para as variáveis. Lembramos que uma cláusula em que ocorrem x_j e $\overline{x}_j, j \in [n]$, pode ser desconsiderada, visto que sempre é verdadeira.

Vamos agora reduzir **3-SAT 2 em 3** para **AGD**(G, D). Dada uma fórmula como entrada para **3-SAT 2 em 3**, vamos construir o grafo G. Inicialmente, G tem um vértice universal v. Ver Figura 3.6 para acompanhar a construção. Para cada variável x, v está conectado a vértices v_x^1 e v_x^2 por arestas e_x e $e_{\overline{x}}$, respectivamente. v_x^1 e v_x^2 estão conectados por uma aresta a_x. Para cada cláusula $C = l_1 \vee l_2 \vee l_3$, v estará conectado a vértices v_C^1 e v_C^2 por arestas $e_{l_1}^C$ e $e_{l_2}^C$, respectivamente. Por fim, há uma aresta $e_{l_3}^C$ entre v_C^1 e v_C^2.

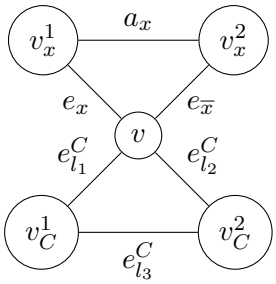


Figura 3.6: Ilustração da construção do grafo G, pela redução.

Para a construção do digrafo D, haverá arcos de a_x para e_x e $e_{\overline{x}}$, para toda variável x. Existirá um arco de e_l para e_l^C, se o literal l ocorrer na cláusula C. A Figura 3.7 exemplifica a construção de D, onde supomos que o literal x aparece nas cláusulas C_i e C_k e que $\overline{x}$ ocorre em C_j.

Observe que os arcos de a_x para e_x e $e_{\overline{x}}$ garantem a exclusão mútua entre e_x e $e_{\overline{x}}$ já que, na árvore, a escolha de ambas implicaria a de a_x, pelas dependências, e um triângulo

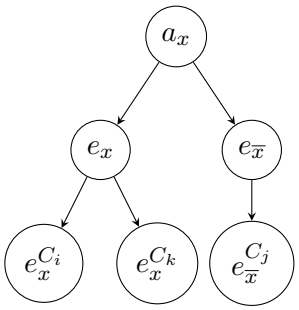


Figura 3.7: Ilustração da construção do digrafo D, pela redução.

seria formado. Observe também que o grafo G é um cacto cordal, visto que todos os seus ciclos são triângulos e têm em comum apenas o vértice v, e que o digrafo D é uma união de arborescências.

Note que, em G, o número de vértices é $2n+2m+1$ e o número de arestas é $3n+3m$, onde n é o número de variáveis da instância de **3-SAT 2 em 3** e m o número de suas cláusulas. Além disso, o digrafo D possui $3n + 3m$ vértices e $2n + 3m$ arcos. Vemos, portanto, que G e D podem ser construídos em tempo polinomial, em função de n e m.

Proposição 3.2. ***AGD***(G, D) *é* ***NP-completo****, mesmo que G seja um cacto cordal com diâmetro 2 e D seja uma união de arborescências de altura 2.*

Demonstração. Primeiro, notamos que **AGD**(G, D) está em **NP**, pois dado um subconjunto de arestas de G, podemos verificar em tempo polinomial se tal subconjunto induz uma árvore geradora de G e satisfaz as restrições de dependência impostas por D. Vamos agora mostrar a validade da redução acima apresentada. Primeiro, tome uma valoração que torna a instância de **3-SAT 2 em 3** verdadeira, com exatamente dois literais verdadeiros por cláusula. Considere o subgrafo T de G induzido pelas seguintes arestas: a_x, para toda variável x; ou e_x ou $e_{\overline{x}}$, dependendo se a valoração torna a variável x verdadeira ou falsa, respectivamente; para cada cláusula $C = l_1 \vee l_2 \vee l_3$, as arestas $e_{l_i}^C$ e $e_{l_j}^C$ tal que l_i e l_j assumem valor verdadeiro, $1 \leq i, j \leq 3$, $i \neq j$. Observe que as arestas de T satisfazem as dependências impostas por D. Mostramos agora que T é uma árvore.

Para todo x, v está ligado a v_x^1 ou v_x^2 por e_x ou $e_{\overline{x}}$, respectivamente, enquanto a aresta a_x liga v_x^1 a v_x^2. Dessa forma, não há ciclo entre v, v_x^1 e v_x^2, e as duas arestas mantém os três vértices conectados. Para todo $C = l_1 \vee l_2 \vee l_3$, como exatamente duas das arestas $e_{l_1}^C$, $e_{l_2}^C$ e $e_{l_3}^C$ estão em T, temos que v liga-se a v_C^1 ou v_C^2 por $e_{l_1}^C$ ou $e_{l_2}^C$, respectivamente. Sem perda de generalidade, suponha que v liga-se a v_C^1 por $e_{l_1}^C$. Se $e_{l_2}^C$ está em T, temos que v_C^2 também

liga-se a v; caso contrário, temos que v_C^2 liga-se a v_C^1 por $e_{l_3}^C$. Em ambos os casos, não há ciclo entre v, v_C^1 e v_C^2, e as duas arestas mantém os três vértices conectados. Sendo assim, v e os vértices associados a cada variável induzem um subgrafo conexo e acíclico de T. O mesmo acontece com os vértices associados a cada cláusula. Logo, T é conexo e acíclico e portanto uma árvore geradora.

Agora, suponha que $T(V, E')$ seja uma solução viável para $\mathbf{AGD}(G, D)$. Temos que, para toda variável x, ou a aresta e_x ou a aresta $e_{\overline{x}}$ estão em T, caso contrário v_x^1 e v_x^2 não se conectariam a v (se ambas não estivessem) ou formariam um ciclo com v (se ambas estivessem, pois a_x também é escolhida por ser dependência de ambas). Temos também que, para toda cláusula $C = l_1 \vee l_2 \vee l_3$, exatamente duas das arestas $e_{l_1}^C, e_{l_2}^C$ e $e_{l_3}^C$ estão em T para conectar v_C^1 e v_C^2 a v e não formar um ciclo com esse. Além disso, e_l^C só pode estar em T se e_l também estiver. Portanto, atribuímos a x valor verdadeiro ($e_x \in E'$) ou falso ($e_{\overline{x}} \in E'$), e esta valoração satisfaz toda cláusula C com exatamente dois literais verdadeiros. Assim, decidimos que a instância de **3-SAT 2 em 3** é satisfatível. □

Note, por fim, que podemos transformar D, uma união de arborescências, em uma arborescência, sem prejuízo para a redução. Basta criar uma folha artificial em G conectada, digamos, a v e fazer sua única aresta adjacente ser dependência para as raízes das arborescências de D. Como é de corte, tal aresta sempre é presente em toda solução viável, e portanto as fontes originais de D têm sua única dependência satisfeita, o que deixa intacto o espaço viável do problema.

3.4.2 Redução para bipartidos

Nesta seção, apresentamos uma outra redução, de **3-SAT** para $\mathbf{AGD}(G, D)$. Apesar de já demonstrada a **NP-completude** de $\mathbf{AGD}(G, D)$, fazemos esta outra redução para mostrar que o problema continua difícil mesmo quando as relações de dependência ocorrem apenas entre arestas com um vértice em comum, ou seja, quando o grafo subjacente a D seja subgrafo de G_L.

Dada uma fórmula como entrada para **3-SAT**, vamos construir o grafo $G(V, E)$. G tem um vértice v e, para cada variável x da fórmula de entrada, tem vértices v_x^1, v_x^2 e v_x^3. Há uma aresta a_x entre v_x^1 e v_x^3 e uma aresta $a_{\overline{x}}$ entre v_x^2 e v_x^3, assim como uma aresta e_x entre v_x^1 e v e outra $e_{\overline{x}}$ entre v_x^2 e v. Para cada cláusula $C = l_1 \vee l_2 \vee l_3$, há um vértice v_C e vértices $v_{l_1}^C$, $v_{l_2}^C$ e $v_{l_3}^C$. Há uma aresta $e_{l_i}^C$ entre $v_{l_i}^C$ e v, para $1 \leq i \leq 3$, assim como uma aresta $a_{l_i}^C$ entre $v_{l_i}^C$ e v_C, para $1 \leq i \leq 3$. Isso conclui a construção do grafo G, que é ilustrada na Figura 3.8.

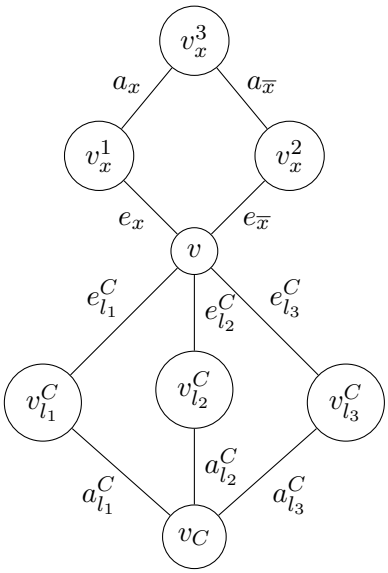


Figura 3.8: Ilustração da construção do grafo G, pela segunda redução.

Agora vamos construir o digrafo $D(E, A)$. Há um arco de a_x para e_x, assim como de $a_{\overline{x}}$ para $e_{\overline{x}}$, para toda variável x da fórmula de entrada. Há também um arco de e_l para e_l^C se o literal l ocorre na cláusula C. Os vértices $a_{l_i}^C$, para $1 \leq i \leq 3$, para toda cláusula C, ficam isolados. A construção do digrafo D é ilustrada na Figura 3.9, onde supomos que o literal x aparece nas cláusulas C_i e C_k e que $\overline{x}$ ocorre em C_j.

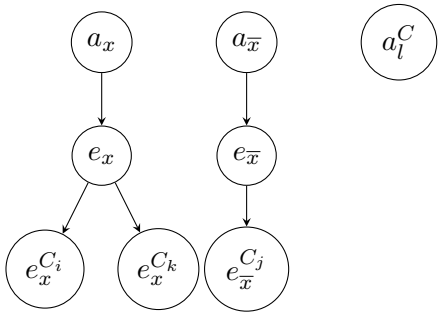


Figura 3.9: Ilustração da construção do digrafo D, pela segunda redução.

Note que e_x e $e_{\overline{x}}$ não podem ambas fazer parte de uma mesma solução, pois elas, junto com suas dependências, formariam um ciclo. Ambas também não podem estar ausentes de uma solução, visto que elas formam um corte que separa v_x^1, v_x^2 e v_x^3 dos demais vértices de G. Assim, temos que exatamente uma delas faz parte de uma dada solução. Note também que G é um grafo bipartido, visto que todo ciclo presente em G tem comprimento 4. O digrafo D é, mais uma vez, uma união de arborescências.

Note que o número de vértices de G é $3n + 4m + 1$, e seu número de arestas é $4n + 6m$, onde n é o número de variáveis da fórmula de entrada para **3-SAT** e m o número de suas cláusulas. Além disso, o digrafo D possui $4n+6m$ vértices e $2n+3m$ arcos. Vemos, portanto, que G e D podem ser construídos em tempo polinomial, em função de n e m.

Proposição 3.3. ***AGD***(G, D) *é* ***NP-completo****, mesmo que as relações de dependência ocorram apenas entre arestas adjacentes, G seja um grafo bipartido e D seja uma união de arborescências de altura 2 e de vértices isolados.*

Demonstração. Mostramos anteriormente que **AGD**(G, D) está em **NP**. Vamos agora validar a redução acima apresentada. Tome uma fórmula de entrada para **3-SAT**. Suponha que existe uma valoração para suas variáveis que a satisfaz. Para cada variável x, escolhemos a_x ou $a_{\overline{x}}$, e então ou e_x ou $e_{\overline{x}}$, dependendo do valor atribuído a x pela valoração. Para uma cláusula $C = l_1 \vee l_2 \vee l_3$ qualquer, C tem pelo menos um literal verdadeiro. Suponha sem perda de generalidade que esse seja l_1. Temos que a aresta $e^C_{l_1}$ tem sua dependência satisfeita, então a usamos para conectar v a $v^C_{l_1}$. Usamos as arestas $a^C_{l_i}$ para conectar v_C a $v^C_{l_i}$, para todo $1 \leq i \leq 3, i \in \mathbb{N}$. Note que existe um caminho entre qualquer vértice acima citado e v, e que as arestas usadas não formam ciclos. Portanto, quando há uma solução para a fórmula de entrada de **3-SAT**, há uma solução para **AGD**(G, D).

Agora, suponha que $T(V, E')$ seja uma solução viável para **AGD**(G, D). Como as arestas e_x e $e_{\overline{x}}$ dependem de a_x e $a_{\overline{x}}$, respectivamente, exatamente uma de e_x e $e_{\overline{x}}$ está em T, já que T é conexo e acíclico. Assim, criamos uma valoração em que x tem valor verdadeiro se e_x está em T ou tem valor falso se $e_{\overline{x}}$ está em T. Para cada cláusula C, temos que as arestas $e^C_{l_i}, 1 \leq i \leq 3$, formam um corte em G. Como T é viável, pelo menos uma delas faz parte de T e o seu respectivo literal é feito verdadeiro pela valoração, de acordo com as relações de dependência. Assim, vemos que a valoração construída satisfaz a instância de **3-SAT**. □

3.5 Variações do problema

Nessa seção, definimos dois outros problemas, **AGD-t**(G, D) e **AGD-1**(G, D), também relacionados com árvores geradoras e restrições de dependência. Em **AGD**(G, D), uma aresta pode compor a solução se *pelo menos* uma de suas dependências é também escolhida. Nesses dois outros problemas, pedimos que **todas** as ou **exatamente uma** de suas dependências sejam escolhidas, respectivamente. Com as mesmas reduções descritas acima, verificamos que essas variantes também são **NP-completas**.

3.5.1 AGD-t(G, D)

Definimos **AGD-t**(G, D), um problema de decisão, como segue.

Problema 3.3. *Seja $G(V, E)$ um grafo e $D(E, A)$ um digrafo cujos vértices são as arestas de G. O problema da Árvore Geradora com Dependências Totais, abreviado como **AGD-t**(G, D), consiste em decidir se existe uma árvore geradora $T(V, E')$ de G tal que $e \in E'$ implica que e é uma fonte em D ou $e' \in E'$ para todo $(e', e) \in A$. Em outras palavras, uma aresta pode fazer parte de uma solução apenas se ela não tem dependências ou, caso tenha, se todas elas também fazem parte da solução.*

Note que a diferença entre **AGD**(G, D) e **AGD-t**(G, D) está em como as dependências são exigidas. Enquanto o primeiro determina que cada aresta com dependências pode aparecer em uma solução apenas junto com ao menos uma delas, o segundo determina que cada aresta com dependências pode aparecer em uma solução apenas junto com todas elas.

A seguir, mostramos que as reduções apresentadas nas subseções 3.4.1 e 3.4.2 também se aplicam a **AGD-t**(G, D). Com isso, mostramos que **AGD-t**(G, D) é **NP-completo**, mesmo nas condições estabelecidas pelas proposições 3.2 e 3.3.

Proposição 3.4. ***AGD-t**(G, D) é **NP-completo**, mesmo que G e D tenham as propriedades descritas pela Proposição 3.2 ou pela Proposição 3.3.*

Demonstração. Tome G e D um grafo e um digrafo, respectivamente, construídos a partir de uma dada fórmula para **3-SAT 2 em 3** (resp. **3-SAT**) como descrito na Subseção 3.4.1 (resp. 3.4.2). Pela Proposição 3.2 (resp. 3.3), temos que a fórmula de **3-SAT 2 em 3** (resp. **3-SAT**) é satisfazível se, e somente se, **AGD**(G, D) tem solução. Note que toda aresta de G ou é uma fonte em D ou tem exatamente uma dependência. Dessa forma, podemos ver que os conjuntos de soluções de **AGD**(G, D) e **AGD-t**(G, D) são iguais. Como implicação da Proposição 3.2 (resp. 3.3), a fórmula de **3-SAT 2 em 3** (resp. **3-SAT**) é satisfazível se e somente se **AGD-t**(G, D) tem solução, e portanto **AGD-t**(G, D) é **NP-completo**. □

3.5.2 AGD-1(G, D)

Definimos também **AGD-1**(G, D), outro problema de decisão, como segue.

Problema 3.4. *Seja $G(V, E)$ um grafo e $D(E, A)$ um digrafo cujos vértices são as arestas de G. O problema da Árvore Geradora com Dependência Única, abreviado como **AGD-1**(G, D), consiste em decidir se existe uma árvore geradora $T(V, E')$ de G tal que $e \in E'$*

implica que e é uma fonte em D ou $e' \in E'$ para exatamente um arco da forma $(e', e) \in A$. Em outras palavras, uma aresta pode fazer parte de uma solução apenas se ela não tem dependências ou, caso tenha, se exatamente uma delas também faz parte da solução.

Agora, provamos que **AGD-1**(G, D) também é **NP-completo**.

Proposição 3.5. ***AGD-1***(G, D) *é* ***NP-completo****, mesmo que G e D tenham as propriedades descritas pela Proposição 3.2 ou pela Proposição 3.3.*

Demonstração. Assim como na demonstração da Proposição 3.4, notamos que G e D, como foram construídos, são tais que toda aresta com dependências a cumprir tem exatamente uma dependência. Isso faz com que as restrições de dependência de **AGD-t**(G, D) sejam equivalentes às de **AGD-1**(G, D). Assim, **AGD-t**(G, D) tem solução se, e somente se, **AGD-1**(G, D) tem solução, e portanto **AGD-1**(G, D) é **NP-completo**. □

3.6 Propriedades e casos polinomiais

Nesta seção, expomos algumas propriedades de **AGDM**(G, D, w) com a finalidade de explorar as características do problema, bem como algumas de suas consequências. Centralizamos nossa análise a partir da estrutura do digrafo D. Particularmente, consideramos casos em que D é a união de estrelas e ciclos orientados ou D é uma arborescência cujo grafo subjacente é caterpílar.

As restrições de dependência de **AGDM**(G, D, w) são representadas por arcos no digrafo $D(E, A)$. Portanto, uma instância do **AGDM**(G, D, w) sem restrições de dependência tem um digrafo vazio como D, isto é, $A = \emptyset$.

Propriedade 3.1. *Se D for um digrafo vazio,* ***AGDM***(G, D, w) *se reduz a* ***AGM***(G, w).

Na Seção 3.4, mostramos que o problema é **NP-difícil** mesmo quando D é uma união de arborescências de altura 2. Quando a altura de uma arborescência for um, a arborescência é uma estrela direcionada. Caso D seja uma estrela direcionada de centro e, e é uma aresta independente e toda outra aresta de G depende unicamente de e. Assim, vê-se que toda solução para **AGDM**(G, D, w) conterá a aresta e.

Propriedade 3.2. *Se D for uma estrela direcionada centrada em e,* ***AGDM***(G, D, w) *reduz-se a* ***AGM***(G', w)*, onde G' é obtido a partir de G pela contração de e.*

Se D for um ciclo orientado, podemos notar que, por transitividade, uma aresta qualquer de G depende de todas as outras. Ou seja, é preciso que todas as arestas de G façam parte

da solução para que toda restrição de dependência seja satisfeita. Desse modo, $\mathbf{AGD}(G, D)$ é viável apenas se G é uma árvore, e portanto o próprio G é a única solução possível para o problema.

Propriedade 3.3. *Se D for um ciclo orientado, $\boldsymbol{AGD}(G, D)$ é viável se e somente se G for uma árvore.*

Agora tome D um digrafo com $k = O(\log_2 n(G))$ componentes, em que cada componente é uma estrela direcionada ou um ciclo orientado. Para cada componente, podemos escolher se suas arestas farão, ou não, parte da construção de uma solução:

- Se a componente for uma estrela direcionada, escolheremos incluir ou não sua aresta central (no primeiro caso, podemos incluir qualquer subconjunto de suas outras arestas; no segundo, nenhuma de suas arestas pode ser selecionada).
- Se a componente for um ciclo direcionado, escolheremos incluir todas as suas arestas ou incluir nenhuma.

Como D tem k componentes, temos 2^k casos possíveis a considerar, ou seja, no máximo $O(n(G))$ casos, já que $k = O(\log_2(n(G)))$. Além disso, cada caso pode ser resolvido em tempo polinomial. Se o grafo sem as arestas excluídas continua conexo e o subgrafo induzido pelas arestas escolhidas é acíclico, o problema se reduz a encontrar uma árvore geradora mínima usando as arestas previamente selecionadas, o que pode ser feito em tempo polinomial [8] [12], mais precisamente $O(n(m + n\log_2(n)))$, com $n = n(G)$ e $m = m(G)$, quando Prim [12] utiliza um *heap* de Fibonacci [5]. Dessa forma, temos a seguinte propriedade.

Propriedade 3.4. *Se D tiver $O(\log_2 n(G))$ componentes, e cada componente for uma estrela direcionada ou um ciclo orientado, $\boldsymbol{AGDM}(G, D, w)$ poderá ser resolvido em tempo polinomial, mais precisamente $O(n(m + n\log_2(n)))$, com $n = n(G)$ e $m = m(G)$.*

Caso D seja um caminho orientado, existe exatamente uma aresta que não tem dependências. A partir dela, seguindo os arcos de D, as $n(G) - 1$ primeiras arestas devem constituir a única solução de $\mathbf{AGDM}(G, D, w)$, se elas induzem uma árvore geradora de G. Caso induzam um subgrafo cíclico, o problema é inviável.

Propriedade 3.5. *Se D é um caminho orientado, ou suas $n(G) - 1$ primeiras arestas formam a única solução de $\boldsymbol{AGDM}(G, D, w)$ ou o problema é inviável.*

Generalizando o caso de caminho orientado, consideramos agora que o digrafo de dependências seja uma arborescência obtida pela orientação de um caterpílar. Formalmente, seja $G(V, E)$ um grafo conexo e $G'(E = P \cup L, E')$ um grafo caterpílar, onde P são os vértices que formam um grafo caminho, caso sejam removidas as folhas L.

Propriedade 3.6. ***AGDM***(G, D, w) *pode ser resolvido em tempo polinomial quando* $D(P \cup L, A)$ *for uma arborescência que seja uma orientação de um grafo caterpílar.*

Demonstração. Seja $D(P \cup L, A)$ uma arborescência enraizada em um vértice $p_1 \in P$, tal que D seja uma orientação de G'. Note que p_1 é vizinho de exatamente um ou dois vértices em P:

- No primeiro caso, onde p_1 tem um vizinho em P, tome $P = \{p_1, p_2, \ldots, p_k\}, k = |P|$, de tal forma que hajam arcos de p_i para p_{i+1}, para todo $i \in [k-1]$. Tome também $L_i \subseteq L$ o conjunto de folhas (em D) cujos arcos incidentes chegam de p_i, para $i \in [k]$. Assim, a aresta p_{i+1} depende da aresta p_i, para $i \in [k-1]$, e as arestas em L_i dependem da aresta p_i, para $i \in [k]$. Note que **AGDM**(G, D, w) pode ser resolvido em k subproblemas. O i-ésimo subproblema, $i \in [k]$, considera as soluções que contêm as arestas $p_j, j \in [i]$, e não contêm as arestas $p_j, i+1 \leq j \leq k$. Sua resolução consiste em começar uma execução do algoritmo de Kruskal [8] com as arestas $p_j, j \in [i]$, inclusas, disponibilizando as arestas em $L_j, j \in [i]$, para inclusão. Dessa forma, a solução para **AGDM**(G, D, w) será a solução de menor custo obtida nos k subproblemas. **AGDM**(G, D, w) será inviável caso todos os k subproblemas também o sejam.

- No segundo caso, onde p_1 tem dois vizinhos em P, tome $P = \{p_1, p_1^1, p_2^1, \ldots, p_k^1, p_1^2, p_2^2, \ldots, p_l^2\}, k + l = |P| - 1$, de tal forma que $p_1, p_1^1, \ldots, p_k^1$ e $p_1, p_1^2, \ldots, p_l^2$ sejam caminhos direcionados em D. Tome também $L_1 \subseteq L$ como o conjunto de folhas cujos arcos incidentes chegam de p_1, e $L_i^1, L_j^2 \subseteq L$ os conjuntos analogamente definidos para $p_i^1, i \in [k]$ e $p_j^2, j \in [l]$, respectivamente. Assim, as arestas p_1^1 e p_1^2 dependem da aresta p_1, assim como p_{i+1}^1 e p_{j+1}^2 dependem, respectivamente, das arestas p_i^1 e p_j^2, para $i \in [k-1]$ e $j \in [l-1]$. Da mesma forma, as arestas em L_i^1 e L_j^2 dependem, respectivamente, das arestas p_i^1 e p_j^2, para $i \in [k]$ e $j \in [l]$. Note que **AGDM**(G, D, w) pode ser resolvido em kl subproblemas. Vamos indexar os subproblemas usando tuplas $(i, j), i \in [k], j \in [l]$. O subproblema $(i, j), i \in [k], j \in [l]$, considera as soluções que contêm as arestas p_1, p_a^1 e $p_b^2, a \in [i], b \in [j]$, e exclui as arestas p_a^1 e $p_b^2, i+1 \leq a \leq k, j+1 \leq b \leq l$. Sua resolução consiste em começar

uma execução do algoritmo de Kruskal [8] com as arestas p_1, p_a^1 e $p_b^2, a \in [i], b \in [j]$ inclusas, disponibilizando as arestas em L_a^1 e L_b^2, $a \in [i]$ e $b \in [j]$, para inclusão. Vemos, assim, que a solução para $\mathbf{AGDM}(G, D, w)$ será a solução com custo mínimo dentre as obtidas nos kl subproblemas. $\mathbf{AGDM}(G, D, w)$ será inviável caso todos os kl subproblemas também sejam.

Em ambos os casos, o número de subproblemas a serem resolvidos é polinomial($O(|P|)$ no primeiro caso e $O(|P|^2)$ no segundo) e cada subproblema toma tempo polinomial para ser resolvido, mais precisamente $O(m + n\log_2(n))$, com $n = n(G)$ e $m = m(G)$. Note ainda que, se D fosse enraizado em um vértice de L, a aresta correspondente em G precisaria estar na solução. Então, bastaria contrair tal aresta em G e remover a raiz de D para obter G'' e D'' satisfazendo a hipótese inicial. Dessa forma, temos demonstrado a propriedade. □

Para concluir esta seção, apresentamos casos polinomiais em que as dependências são definidas em função das distâncias entre arestas de G. Primeiro, consideramos que $D(E, A)$ seja uma dupla orientação das arestas do grafo linha de G, $G_L(E, E')$, isto é, $A = \{(e_1, e_2), (e_2, e_1) : \{e_1, e_2\} \in E'\}$. Note que qualquer árvore geradora T de G satisfaz as dependências D. A afirmação é trivial, se $|V| \leq 2$; no caso contrário, segue do fato de que qualquer aresta de T possui uma de suas arestas adjacentes, em G, também compondo T.

Propriedade 3.7. *Seja G um grafo e G_L seu grafo linha. Se D for uma dupla orientação de G_L, então $\mathbf{AGDM}(G, D, w)$ se reduzirá a $\mathbf{AGM}(G, w)$.*

Note que D ser uma dupla orientação de G_L corresponde a pedir que $N^-(e) = \{e' \in E : d(e, e') = 1\}$, onde a distância é tomada em G_L. Analisamos agora o caso em que D possui a seguinte propriedade, com respeito a uma aresta de referência e de G:

P Para cada $e' \in E$, $N^-(e') = \emptyset$ ou $N^-(e') = \{e'' \in E : d(e, e'') = k_{e'}\}$, para algum $1 \leq k_{e'} \leq d(e, e') - 1$.

Em outros termos, toda aresta e' que não é fonte tem como dependências todas as arestas que estão a uma distância k de e, para algum $k < d(e, e')$. Neste caso, temos que uma solução ótima para $\mathbf{AGDM}(G, D, w)$ é uma árvore geradora de G contendo a aresta e de custo mínimo. De fato, note que, em qualquer árvore geradora de G contendo e, se há uma aresta e', deve haver também um conjunto de arestas $\{e_0 = e, e_1, \ldots, e_p\}$, onde $p = d(e, e') - 1$ e $d(e, e_i) = i, i \in [p]$. Logo, qualquer árvore geradora de G contendo e

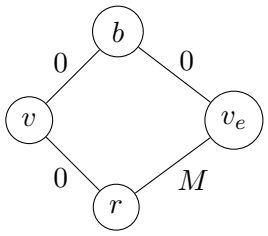


Figura 3.10: Exemplo de construção do grafo G', pela redução. v é um vértice qualquer de G, e v_e representa os vértices criados para cada aresta e de G.

satisfaz as precedências D. Essa observação nos permite identificar as seguintes instâncias fáceis do problema.

Propriedade 3.8. *Sejam $G(V, E)$ e $D(E, A)$ satisfazendo a propriedade **P**. Uma solução para **AGDM**(G, D, w) consiste em uma árvore geradora de G contendo a aresta e de custo mínimo e tal solução pode ser encontrada em tempo polinomial.*

3.7 Viabilidade

Na Seção 3.4, vimos que mesmo determinar uma solução viável para **AGDM**(G, D, w) é um problema **NP-completo**. Isso pode inviabilizar o uso de métodos de resolução que partem de uma solução viável. Nesse sentido, apresentamos nesta seção um modo de reduzir qualquer instância de **AGDM**(G, D, w) a uma de **AGDM**(G', D', w') em que é fácil encontrar uma solução viável e que preserva o custo ótimo, caso a instância original seja viável, ou que tem custo ótimo arbitrariamente grande, caso contrário.

Seja $G(V_1, E_1), D(E_1, A)$ um grafo e digrafo como entradas em **AGDM**(G, D, w). Vamos construir $G'(V_2, E_2), D'(E_2, A')$ e w' como segue. Fazemos $V_2 = V_1 \cup \{b, r\} \cup \{v_e : e \in E_1\}$, assim como $E_2 = E_1 \cup \{\{v_e, b\}, \{v_e, r\} : e \in E_1\} \cup \{\{v, b\}, \{v, r\}\}$, dado um vértice v qualquer de G. Ver Figura 3.10. Para obter D', adicionamos dependências para as arestas que não são fontes em D, fazendo $A' = A \cup \{(\{v_e, r\}, e) : e \in E, N^-(e) \neq \emptyset\}$. A nova função de custo w' estende a anterior da seguinte forma: $w'_e = w_e, \forall e \in E_1; w'_{\{v,b\}} = 0; w'_{\{v,r\}} = 0; w'_{\{v_e,b\}} = 0, \forall e \in E_1; w'_{\{v_e,r\}} = M, \forall e \in E_1$, onde $M \in \mathbb{R}^+$ é um número suficientemente grande. G' e D' são construídos de tal forma que toda aresta $e \in E_1$ que não é fonte em D passa a ter $\{v_e, r\}$ como dependência.

Proposição 3.6. *Sejam $G(V, E)$ um grafo conexo e $D(E, A)$ um digrafo. Sejam G' e D' construídos como descrito acima. Suponha $M > \sum_{e \in E'} w_e$, onde $E' \subseteq E$ é o conjunto das $n(G) - 1$ arestas de maior peso. Então ocorre um, e somente um, dos seguintes casos:*

- ***AGDM***(G, D, w) *e* ***AGDM***(G', D', w') *têm o mesmo valor ótimo.*
- *O valor ótimo de* ***AGDM***(G', D', w') *é maior ou igual a* M *e* ***AGDM***(G, D, w) *é inviável.*

Demonstração. Primeiro note que podemos obter uma solução viável y para **AGDM**(G', D', w') a partir de uma solução x - ambas vetores característicos - qualquer para **AGM**(G, w) da seguinte forma: para todo $e \in E$ com $N^-(e) \neq \emptyset$ e $x_e = 1$, acrescentamos a aresta $\{v_e, b\}$, caso uma dependência de e já esteja em x, ou acrescentamos a aresta $\{v_e, r\}$, caso contrário. Mais precisamente, fazemos $y_e = x_e, \forall e \in E$; $y_{\{v,b\}} = y_{\{v,r\}} = 1$; $y_{\{v_e,b\}} = 1 - y_{\{v_e,r\}}, \forall e \in E$, onde

$$y_{\{v_e,r\}} = \begin{cases} \max(0, 1 - \sum_{e' \in N^-(e)} x_{e'}), \text{ se } N^-(e) \neq \emptyset \\ 0, \text{ caso contrário} \end{cases} \tag{3.1}$$

Logo, **AGDM**(G', D', w') é sempre viável e, portanto, tem solução ótima.

Afirmação 3.1. *Se* ***AGDM***(G, D, w) *for viável, seu valor ótimo será menor que* M *e maior ou igual ao valor ótimo de* ***AGDM***(G', D', w').

Demonstração. Seja x solução ótima para **AGDM**(G, D, w). Então x é viável para **AGM**(G, w) e a solução y construída para **AGDM**(G', D', w') é tal que $y_e = x_e, \forall e \in E; y_{\{v,b\}} = y_{\{v,r\}} = 1; y_{\{v_e,r\}} = 0, \forall e \in E; y_{\{v_e,b\}} = 1, \forall e \in E$. Logo, x e y têm o mesmo custo. Como o custo de x é certamente menor que M, o resultado segue. □

Admita inicialmente que o valor ótimo de **AGDM**(G', D', w') é maior ou igual a M. Pela Afirmação 3.1, concluímos que **AGDM**(G, D, w) é inviável. Por outro lado, se é menor que M, a solução ótima não usa qualquer aresta $\{v_e, r\}$, mas usa todas as arestas $\{v_e, b\}$. Removendo-as, obtemos uma solução viável para **AGDM**(G, D, w) de mesmo valor. Consequentemente, o valor ótimo de **AGDM**(G, D, w) é menor ou igual ao valor ótimo de **AGDM**(G', D', w') e a igualdade segue pela Afirmação 3.1. □

3.8 Problemas relacionados

Nesta seção, procuramos reduzir polinomialmente alguns problemas a **AGDM**(G, D, w), procurando certificar a abrangência deste. Em particular, reduzimos o problema de Árvore Geradora Mínima sob Restrições de Conflito, recentemente estudado na literatura.

3.8.1 Árvore Geradora com Conflitos Mínima

Em [2], é definido o problema de Árvore Geradora Mínima sob Restrições de Conflito, a ser denotado $\mathbf{AGRC}(G, G_c, w)$, que consiste em encontrar uma árvore geradora de $G(V, E)$ que respeite as restrições de conflito impostas por $G_c(E, E_c)$ e que tenha custo mínimo para a função de custo $w : E \to \mathbb{R}^+$. Lembramos aqui que uma aresta $\{e_1, e_2\}$ em G_c indica um conflito entre as arestas e_1 e e_2 de G, ou seja, ambas não podem fazer parte de uma mesma solução viável. Esse problema foi mostrado ser **NP-difícil** em [2]. Recentemente, em [13], foi proposto um método de *branch and bound* para o mesmo.

Vamos descrever uma transformação de $\mathbf{AGRC}(G, G_c, w)$ em $\mathbf{AGDM}(G', D, w')$, mantendo seu custo ótimo. Tomamos G e um vértice qualquer $v \in V$. Para cada aresta $\{e_1, e_2\} \in E_c$, expandimos G da seguinte forma: criamos vértices $v_{e_1e_2}$ e $v_{\overline{e}_1\overline{e}_2}$, além de vértices $v_{e_1\overline{e}_2}$ e $v_{e_2\overline{e}_1}$, assim como arestas $a'_{\overline{e}_1\overline{e}_2} = \{v_{e_1e_2}, v_{\overline{e}_1\overline{e}_2}\}, a_{\overline{e}_1\overline{e}_2} = \{v_{\overline{e}_1\overline{e}_2}, v\}, a'_{e_2\overline{e}_1} = \{v_{e_1e_2}, v_{e_2\overline{e}_1}\}, a'_{e_1\overline{e}_2} = \{v_{e_1e_2}, v_{e_1\overline{e}_2}\}, a_{e_2\overline{e}_1} = \{v_{e_2\overline{e}_1}, v\}$ e $a_{e_1\overline{e}_2} = \{v_{e_1\overline{e}_2}, v\}$, além de arestas $\{v_{\overline{e}_1\overline{e}_2}, v_{e_2\overline{e}_1}\}$ e $\{v_{\overline{e}_1\overline{e}_2}, v_{e_1\overline{e}_2}\}$. Feito isso, temos construído o grafo G'. Ilustramos a construção de G' na Figura 3.11. Criamos o digrafo D, que será uma união de vértices isolados e caminhos orientados, um para cada aresta de G, como segue: para cada aresta $e \in E$, criamos um caminho orientado que passa por todas as arestas da forma $a_{e\overline{f}}$ e $a'_{e\overline{f}}$, onde f é uma aresta qualquer que tem conflito com e, e fazemos seu sumidouro apontar para e; as arestas remanescentes tornam-se vértices isolados em D. Por último, definimos a função w' como $w'(e) = w(e)$, se $e \in E$ e $w'(e) = 0$, caso contrário.

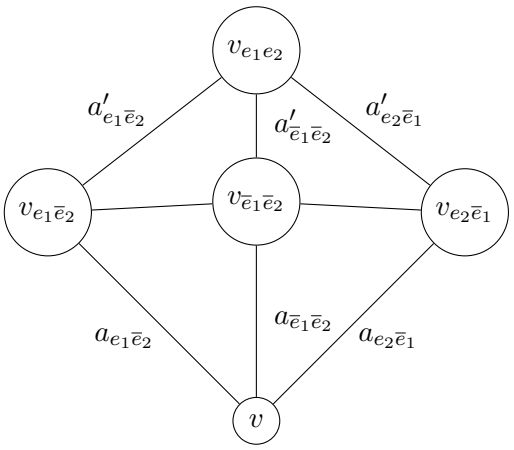


Figura 3.11: Ilustração da construção do grafo G', pela redução.

Proposição 3.7. *$\mathbf{AGDM}(G', D, w')$ e $\mathbf{AGRC}(G, G_c, w)$ têm o mesmo valor ótimo.*

Demonstração. Seja T uma solução viável para $\mathbf{AGRC}(G, G_c, w)$. T deve, portanto, respeitar as restrições de conflito impostas por G_c. Construímos uma solução viável para $\mathbf{AGDM}(G', D, w')$ tomando todas as arestas de T e as arestas necessárias para satisfazer suas dependências, definidas por cada caminho em D. Seja T' o subgrafo gerador de G' com essas arestas. Notamos que as arestas de T' exclusivas de G' induzem ciclo se, e somente se, há conflito entre as arestas que delas dependem. Assim, por ser viável, as arestas de T e suas dependências não formam ciclo em G'. Resta então converter o grafo acíclico T' em uma árvore geradora. Para cada aresta $\{e_1, e_2\} \in E_c$, selecionamos as arestas adjacentes a $v_{\overline{e}_1\overline{e}_2}$ necessárias para, junto com as anteriormente selecionadas, induzir uma árvore de G'. Se e_1 ou e_2 está em T, adicionamos a T' as duas arestas de G' incidentes a $v_{\overline{e}_1\overline{e}_2}$ e incidentes a $v_{e_1\overline{e}_2}$ ou a $v_{e_2\overline{e}_1}$. Se nem e_1 nem e_2 estão em T, adicionamos as quatro arestas de G' incidentes a $v_{\overline{e}_1\overline{e}_2}$. Note ainda que, à exceção das arestas de T, todas as selecionadas para T' têm peso nulo, e portanto T e T' têm o mesmo peso. Tomando T uma solução ótima de $\mathbf{AGRC}(G, G_c, w)$ com custo c, temos que o custo ótimo de $\mathbf{AGDM}(G', D, w')$ é no máximo c.

Tome agora T uma solução viável para $\mathbf{AGDM}(G', D, w')$. Como T é acíclica, temos que não há conflito entre as arestas de T que estão em G. Essas arestas induzem uma árvore de G, visto que G é um subgrafo conexo de G' induzido por V, e portanto são uma solução viável para $\mathbf{AGRC}(G, G_c, w)$. Como as arestas de T que estão em G são suas únicas arestas com pesos positivos, temos que o custo dessa solução viável é o mesmo de T. Tomando T uma solução ótima de $\mathbf{AGDM}(G', D, w')$ com custo c, temos que o custo ótimo de $\mathbf{AGRC}(G, G_c, w)$ é no máximo c. □

3.8.2 Restrições de Empacotamento, Cobertura ou Particionamento

Dado um problema qualquer que envolve escolher um subconjunto de arestas de um grafo $G(V, E)$, uma restrição de empacotamento (cobertura, particionamento) de um conjunto $K \subseteq E$ de arestas exige que no máximo (pelo menos, exatamente) uma aresta em K seja escolhida para a solução. Dada uma família $\mathcal{K} = \{K_1, K_2, \ldots, K_p\}$ de subconjuntos de E, consideramos o Problema de Árvore Geradora Mínima com uma restrição de empacotamento (cobertura, particionamento) para cada $K_i, i = 1, 2, \ldots, p$, a ser denotado $\mathbf{AGME}(G, \mathcal{K}, w)$ ($\mathbf{AGMC}(G, \mathcal{K}, w)$, $\mathbf{AGMP}(G, \mathcal{K}, w)$). Mostramos que esses três problemas podem ser convertidos em $\mathbf{AGDM}(G', D', w')$, para alguma tripla (G', D', w').

Note que $\mathbf{AGME}(G, \mathcal{K}, w)$ é equivalente a $\mathbf{AGRC}(G, G_c, w)$, e portanto pode ser des-

crito por restrições de dependência, como mostrado na Subeção 3.8.1. De fato, a restrição de empacotamento para cada $K \in \mathcal{K}$ é equivalente a um conjunto de restrições de conflito, uma para cada par de elementos de K.

O problema $\mathbf{AGMC}(G, \mathcal{K}, w)$ também pode ser descrito por restrições de dependência. Pode-se determinar que pelo menos uma aresta de $K \in \mathcal{K}$ venha a fazer parte das soluções viáveis da seguinte forma: criamos em G uma folha artificial para cada $K \in \mathcal{K}$ e fazemos sua aresta adjacente depender das arestas de K. O custo dessa aresta seria 0.

A partir do que foi exposto nos parágrafos anteriores, torna-se trivial assegurar que a restrição adicional de $\mathbf{AGMP}(G, \mathcal{K}, w)$ pode ser descrita como dependência. Ora, exigir que haja exatamente uma aresta de cada $K \in \mathcal{K}$ é pedir as duas restrições anteriores sobre K. Basta então implementar conjuntamente as duas reduções.

3.8.3 Restrições de Empacotamento, Cobertura ou Particionamento Condicionadas

Consideramos agora variantes dos problemas definidos na Seção 3.8.2. Dado um problema qualquer que envolve escolher um subconjunto de arestas de um grafo $G(V, E)$, uma restrição de empacotamento (cobertura, particionamento) de um conjunto $K \subset E$ de arestas condicionada a uma aresta $e \in E \setminus K$ exige que, caso e seja escolhida, no máximo (pelo menos, exatamente) uma aresta em K seja escolhida para a solução. Dada uma família $\mathcal{K} = \{(K_1, e_1), (K_2, e_2), \ldots, (K_p, e_p)\}$ de pares, onde $K_i \subset E$ e $e_i \in E \setminus K_i$, para $i \in [p]$, consideramos o Problema de Árvore Geradora Mínima com uma restrição de empacotamento (cobertura, particionamento) condicionada para cada $(K, e) \in \mathcal{K}$, a ser denotado $\mathbf{AGMEC}(G, \mathcal{K}, w)$ ($\mathbf{AGMCC}(G, \mathcal{K}, w)$, $\mathbf{AGMPC}(G, \mathcal{K}, w)$). Note que $\mathbf{AGMCC}(G, \mathcal{K}, w)$ é equivalente a $\mathbf{AGDM}(G, D, w)$, onde D tem arcos de cada aresta de K para e, para todo $(K, e) \in \mathcal{K}$.

Em se tratando de $\mathbf{AGMEC}(G, \mathcal{K}, w)$, conseguimos implementar suas restrições adicionais quando cada conjunto K em $\mathcal{K}$ tem cardinalidade 2. Para cada $(K = \{e_1, e_2\}, e) \in \mathcal{K}$, expandimos $G(V, E)$ da seguinte forma: dado um vértice v qualquer de G, criamos vértices v_K^1, v_K^2, v_K^3 e v_K e arestas $a_e^K = \{v_K^1, v_K^3\}$, $b_e^K = \{v_K^2, v_K^3\}, a_{e_1}^K = \{v, v_K^1\}$, $a_{e_2}^K = \{v, v_K^2\}$, além de arestas $\{v_K, v_K^1\}, \{v_K, v_K^2\}, \{v_K, v_K^3\}$ e $\{v, v_K\}$. O custo de todas as arestas criadas é 0. Ver Figura 3.12. Construímos D da seguinte forma: para cada aresta $e \in E$, criamos um caminho orientado que passa por toda aresta das formas a_e^K e b_e^K, e fazemos o sumidouro desse caminho apontar para e. Assim, para $(K = \{e_1, e_2\}, e) \in \mathcal{K}$, quando e é selecionada, temos selecionada também a_e^K, e podemos então selecionar no máximo uma dentre e_1 e e_2,

pois cada uma delas faz selecionar $a^K_{e_1}$ e $a^K_{e_2}$, respectivamente, e essas duas, junto com a^K_e, formariam um ciclo; na ausência de e, podemos selecionar ambas e_1 e e_2; caso nem e_1 nem e_2 sejam selecionadas, fazemos uso das arestas adjacentes a v_K para manter a conectividade.

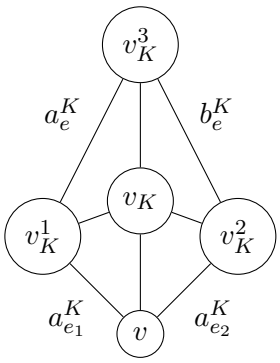


Figura 3.12: Ilustração da expansão do grafo G.

Com isso, podemos implementar restrições de conflito condicionadas para qualquer (K, e) da seguinte forma: caso $|K| > 2$, implementamos uma restrição de conflito condicionada para cada subconjunto de K com cardinalidade 2, como descrito acima. Dessa forma, vemos que **AGMEC**$(G, \mathcal{K}, w)$ pode ser implementado por restrições de dependência.

Já **AGMPC**$(G, \mathcal{K}, w)$ pode ser visto como um problema que impõe restrições de empacotamento e cobertura condicionados para todo $(K, e) \in \mathcal{K}$. Assim, basta implementar essas duas restrições, como é descrito anteriormente, para cada $(K, e) \in \mathcal{K}$, desde que $e_i, i \in [p]$, não pertença a qualquer $K_j, j \in [p]$, para $\mathcal{K} = \{(K_1, e_1), (K_2, e_2), \ldots, (K_p, e_p)\}$.

Capítulo 4

Formulações

Neste capítulo, fazemos a análise de formulações para $\mathbf{AGDM}(G, D, w)$, algumas das quais surgem naturalmente a partir de formulações bastante conhecidas para $\mathbf{AGM}(G, w)$. Por esse motivo, listamos com certo detalhe algumas destas, a fim de melhor compreender as propriedades das formulações que delas derivam.

4.1 Formulações para $\mathbf{AGM}(G, w)$

Nesta seção, apresentamos algumas das formulações mais populares para $\mathbf{AGM}(G, w)$. Todas elas exigem o cumprimento de duas dentre as três condições a seguir, o que garante a obtenção de uma árvore geradora T de um grafo G:

- $n(T) = n(G)$ e $m(T) = n(G) - 1$.
- T é conexo.
- T é acíclico.

Descrevemos uma formulação SEC, baseada em restrições para eliminação de sub-rotas. Além dessa, apresentamos as versões direcionadas de SEC e CUT, respectivamente tratadas como $DSEC$ e $DCUT$. CUT, que deixamos de apresentar por não ter solução relaxada inteira, baseia-se em restrições de corte. Todas essas formulações são encontradas em [9]. Exibimos também uma formulação baseada em desigualdades Miller-Tucker-Zemlin [10], visto que tais restrições são amplamente utilizadas em formulações para problemas envolvendo eliminação de subrotas [14] [7] [1].

SEC, $DSEC$ e $DCUT$ possuem a propriedade da integralidade [9]. Ou seja, sua relaxação linear fornece o valor ótimo inteiro. Entretanto, cada uma dessas formulações possui

um número exponencial de restrições, o que certamente torna o processo de solução dispendioso. Por outro lado, MTZ emprega um número polinomial de variáveis e restrições, sendo uma formulação mais leve. Porém, sua relaxação linear é fraca.

4.1.1 SEC

Dado um grafo $G(V, E)$ e uma função de peso para as arestas de G, $w : E \to \mathbb{R}$, formulamos SEC como segue. Definimos a variável $x_e \in \mathbb{B}, \forall e \in E$, para representar a pertinência da aresta e à solução ótima de SEC, ou seja:

$$x_e = \begin{cases} 1, & \text{se } e \text{ pertence à solução;} \\ 0, & \text{se } e \text{ não pertence à solução.} \end{cases} \tag{4.1}$$

A formulação SEC é então:

$$\min \quad \sum_{e \in E} w_e x_e \tag{4.2}$$

$$\text{s.a} \quad \sum_{e \in E} x_e = n(G) - 1 \tag{4.3}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \qquad \forall S \subset V : |S| \geq 2 \tag{4.4}$$

$$x \in \mathbb{B}^{m(G)} \tag{4.5}$$

A restrição (4.3) garante que a solução ótima terá $n(G)-1$ arestas, enquanto as restrições (4.4) asseguram que nenhum subconjunto S de V induz um ciclo em uma solução viável. Essas propriedades caracterizam a solução ótima como uma árvore geradora de G e, dada a direção de otimalidade, com peso mínimo.

Observe que existe um número exponencial, em relação a $n(G)$, de restrições (4.4). Isso torna inviável resolver essa formulação diretamente, levando-nos a procurar outras maneiras de abordá-la. Uma delas seria gerando as restrições como planos de corte, já que dessa forma elas são incluídas no modelo apenas se violadas. Esse método de resolução é detalhadamente apresentado em [9], onde é mostrado também que sua execução toma tempo polinomial para a formulação acima.

4.1.2 $DSEC$

Dado um grafo $G(V, E)$ e uma função de peso para as arestas de G, $w : E \to \mathbb{R}$, criamos um digrafo $D(V, A)$, onde $A = \{(u, v), (v, u) : \{u, v\} \in E\}$, e selecionamos um vértice $r \in V$ qualquer como raiz. Uma árvore geradora de G pode ser vista como uma arborescência em

D, com raiz r. Assim, além das variáveis $x_{uv}, \forall\{u, v\} \in E$, anteriormente descritas, para definir o modelo $DSEC$, usamos as variáveis $y_{uv}, \forall(u, v) \in A$, com o seguinte significado:

$$y_{uv} = \begin{cases} 1, & \text{se o arco } (u, v) \in A \text{ pertence à solução;} \\ 0, & \text{caso contrário.} \end{cases} \tag{4.6}$$

A formulação é descrita como:

$$\min \quad \sum_{e \in E} w_e x_e \tag{4.7}$$

$$\text{s.a} \quad x_e = y_{uv} + y_{vu}, \qquad \forall e = \{u, v\} \in E \tag{4.8}$$

$$\sum_{(u,v) \in A} y_{uv} = n(G) - 1 \tag{4.9}$$

$$\sum_{u \in N(v)} y_{uv} = 1, \qquad \forall v \in V \setminus \{r\} \tag{4.10}$$

$$\sum_{(u,v) \in A(S)} y_{uv} \leq |S| - 1, \qquad \forall S \subset V : |S| \geq 2 \tag{4.11}$$

$$y \in \mathbb{B}^{m(D)} \tag{4.12}$$

$$x \in \mathbb{B}^{m(G)} \tag{4.13}$$

No modelo acima, a restrição (4.9) garante que exatamente $n(G)-1$ arcos são selecionados para a solução. Já as restrições (4.10) asseguram que haverá um arco chegando em todo vértice com exceção de r, o que, junto com a restrição anterior, faz com que arcos chegando em r não sejam selecionados. Por último, as restrições (4.11) não permitem que mais de $|S| - 1$ arcos sejam selecionados dentre aqueles induzidos por S, determinando que a solução será acíclica, mesmo quando desconsideradas as direções dos arcos. Assim, a solução terá $n(G) - 1$ arcos e será acíclica, o que a caracteriza como uma árvore geradora. Isso, junto com a direção de otimalidade, faz com que a solução seja uma árvore geradora mínima.

4.1.3 *DCUT*

Dado um grafo $G(V, E)$ e uma função de peso para as arestas de G, $w : E \to \mathbb{R}$, construímos o digrafo $D(V, A)$ como feito anteriormente. Além disso, selecionamos um vértice $r \in V$ qualquer como raiz. Usamos também variáveis $y_{uv}, \forall(u, v) \in A$, e variáveis

$x_{uv}, \forall \{u, v\} \in E$ para definir a formulação $DCUT$:

$$\min \quad \sum_{e \in E} w_e x_e \tag{4.14}$$

$$\text{s.a} \quad x_e = y_{uv} + y_{vu}, \qquad \forall e = \{u, v\} \in E \tag{4.15}$$

$$\sum_{(u,v) \in A} y_{uv} = n(G) - 1 \tag{4.16}$$

$$\sum_{(u,v) \in \delta^+(S)} y_{uv} \geq 1, \qquad \forall S \subset V : r \in S \tag{4.17}$$

$$y \in \mathbb{B}^{m(D)} \tag{4.18}$$

$$x \in \mathbb{B}^{m(G)} \tag{4.19}$$

No modelo acima, a restrição (4.16) garante que exatamente $n(G) - 1$ arcos serão selecionados. Já a restrição (4.17) faz com que sempre haja um arco entre quaisquer subconjuntos complementares em V, assegurando que o grafo obtido seja conexo. Essas duas restrições garantem que as arestas selecionadas induzam uma árvore geradora, que terá custo mínimo, devido à direção de otimalidade.

4.1.4 MTZ

Dado um grafo $G(V, E)$ e uma função de peso para as arestas de G, $w : E \to \mathbb{R}$, construímos um digrafo $D(V, A)$ como anteriormente. Selecionamos um vértice $r \in V$ qualquer como raiz. Usamos as variáveis $y_{uv}, \forall (u, v) \in A$, $x_{uv}, \forall \{u, v\} \in E$, já definidas para as formulações anteriores, e ainda variáveis $l_v \in \mathbb{R}, \forall v \in V$, que representam um rótulo para cada vértice. O modelo MTZ segue:

$$\min \quad \sum_{e \in E} w_e x_e \tag{4.20}$$

$$\text{s.a} \quad x_e = y_{uv} + y_{vu}, \qquad \forall e = \{u, v\} \in E \tag{4.21}$$

$$\sum_{(u,v) \in A} y_{uv} = n(G) - 1 \tag{4.22}$$

$$\sum_{u \in N(v)} y_{uv} = 1, \qquad \forall v \in V \setminus \{r\} \tag{4.23}$$

$$l_r = 1 \tag{4.24}$$

$$l_u - l_v + 1 \leq (n(G) - d(r, v))(1 - y_{uv}), \qquad \forall (u, v) \in A, v \neq r \tag{4.25}$$

$$1 + d(r, v) \leq l_v \leq n(G), \qquad \forall v \in V \tag{4.26}$$

$$y \in \mathbb{B}^{m(D)} \tag{4.27}$$

$$x \in \mathbb{B}^{m(G)} \tag{4.28}$$

No modelo apresentado, a restrição (4.22) assegura a seleção de exatamente $n(G) - 1$ arcos. Já a restrição (4.23) diz que há um arco chegando em todo vértice que não seja r. A restrição (4.24) fixa o rótulo de r em 1, enquanto a restrição (4.25) garante que existe um arco de u para v na arborescência apenas se o rótulo de v é estritamente maior que o de u, o que evita a ocorrência de ciclos. Por outro lado, se o arco (u, v) não é escolhido, a restrição resultante em (4.25) torna-se redundante. De fato, por (4.26), temos que $l_u - l_v + 1 \leq n(G) - (1 + d(r, v)) + 1 = n(G) - d(r, v)$. Esse princípio de eliminação de subrotas foi proposto, primeiramente, para o problema do Caixeiro Viajante, em [10]. Vale destacar que, em [10], a restrição (4.25) usa $n(G)$ em lugar do coeficiente $n(G) - d(r, v)$. Note, portanto, que (4.25) é um fortalecimento da restrição original de [10].

Devemos chamar atenção aqui para o fato de que, dentre as formulações para árvore geradora apresentadas, MTZ é a única cujas soluções básicas ótimas da relaxação linear não são inteiras. Podemos ilustrar isso com o grafo a seguir.

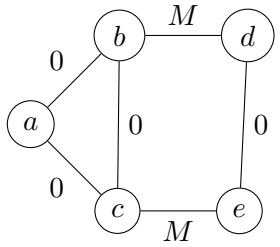


Figura 4.1: Caso patológico para MTZ.

Para o grafo da Figura 4.1, temos que sua árvore geradora mínima deve ter peso M. Uma solução ótima para o modelo MTZ com a versão original de (4.25), associado a esse grafo com raiz em d teria, no entanto, custo 0. Como exemplo, podemos ter $y_{de} = 1, y_{ab} = y_{bc} = y_{ca} = k, y_{ba} = y_{cb} = y_{ac} = 1 - k$, com as demais variáveis y assumindo valor 0 e todas as variáveis l assumindo valor 1, com exceção de $l_e = 2$. Isso nos mostra que a diferença entre o valor ótimo de MTZ e o de sua relaxação linear pode ser arbitrariamente grande.

Há diversas abordagens para fortalecer a restrição original (4.25). Podemos substituí-la, por exemplo, pela desigualdade mais forte apresentada em [4] (com uma leve adaptação, já que aqui tratamos de árvores), que segue, obtida por *lifting*:

$$l_u - l_v + y_{uv} - y_{vu} + (n(G) - 1)(y_{uv} + y_{vu} - 1) \leq 0, \quad \forall (u, v) \in A, u, v \neq r \tag{4.29}$$

Essas desigualdades nos dizem: caso $y_{uv} = 1$, temos $l_v = l_u + 1$; caso $y_{vu} = 1$, temos $l_u = l_v + 1$; caso $y_{uv} = y_{vu} = 0$, $l_u - l_v$ e $l_v - l_u$ podem ser no máximo $n(G) - 1$. Como

vimos, por (4.26), esse valor é de fato limite superior para as diferenças.

Assim como a restrição (4.29) é um *lifting* da versão original de (4.25), temos a seguinte restrição como um *lifting* de (4.25):

$$l_u - l_v + y_{uv} - y_{vu} + (n(G) - d(r, v) - 1)(y_{uv} + y_{vu} - 1) \leq 0, \quad \forall (u, v) \in A, u, v \neq r \quad (4.30)$$

Temos ainda, vindo de [1], um outro fortalecimento para a restrição original (4.25), apresentado a seguir:

$$l_u - l_v + y_{uv} - 2y_{vu} + (n(G) - 1)(y_{uv} + y_{vu} - 1) + \sum_{\substack{k=1, \\ k \neq u}}^{n(G)} y_{kv} \leq 0, \quad \forall (u, v) \in A, u, v \neq r \quad (4.31)$$

Note que $\sum_{\substack{k=1, \\ k \neq u}}^{n(G)} y_{kv} = 1 - y_{uv}$, e então podemos escrever (4.31) como:

$$l_u - l_v + y_{uv} - y_{vu} + (n(G) - 2)(y_{uv} + y_{vu} - 1) \leq 0, \quad \forall (u, v) \in A, u, v \neq r \quad (4.32)$$

Claramente, (4.32) é um *lifting* de (4.29), já que $y_{uv} + y_{vu} \leq 1$. O efeito sobre os rótulos é idêntico àquele gerado por (4.29) quando $y_{uv} = 1$ ou $y_{vu} = 1$. Já quando $y_{uv} = y_{vu} = 0$, o limite superior gerado para a diferença $l_u - l_v$ e para $l_v - l_u$ é uma unidade menor. Como $u, v \neq r$, a diferença máxima entre l_u e l_v (ou l_v e l_u) é $n(G) - 2$.

4.2 Formulação para AGDM(G, D, w)

Para modelar problemas que procuram determinar árvore geradora de um grafo com alguma propriedade desejada, uma estratégia natural é tomar como base uma formulação para **AGM**(G, w) e tentar fazer cumprir as demais propriedades com restrições adicionais. Um critério bem aceito para escolha da formulação base é a qualidade de sua relaxação linear. Neste aspecto, SEC, $DSEC$ e $DCUT$ seriam candidatas naturais. Por outro lado, formulações compactas - com poucas variáveis e restrições - para **AGM**(G, w), mas com relaxação linear fraca, podem se mostrar efetivas para o problema em questão, o que justifica a exposição de MTZ na Seção 4.1.

Apresentamos, pois, um modo de formular **AGDM**(G, D, w) a partir das formulações da Seção 4.1. Definimos, antes de tudo, seus poliedros.

Seja P_{SEC} o poliedro descrito pelas restrições (4.3)-(4.4). Seja também P_{DSEC} o poliedro descrito pelas restrições (4.8)-(4.11). Definimos também P_{DCUT} como o poliedro descrito pelas restrições (4.15)-(4.17). Por último, definimos P_{MTZ} como o poliedro descrito pelas

restrições (4.21)-(4.26). Em todos esses poliedros, consideramos ainda as restrições de não negatividade das variáveis. Em outras palavras, cada um deles define a relaxação linear do modelo correspondente.

As restrições de dependência, que caracterizam **AGDM**(G, D, w), podem ser modeladas linearmente como:

$$\sum_{e' \in N^-(e)} x_{e'} \geq x_e, \quad \forall e \in E : N^-(e) \neq \emptyset \tag{4.33}$$

Podemos notar que a restrição (4.33) determina que uma aresta que não é fonte em D faz parte da solução apenas se ao menos uma das arestas que a precedem também faz parte. Isso assegura que as dependências impostas por D são satisfeitas.

Com isso, criamos formulações para **AGDM**(G, D, w) a partir das enunciadas na Seção 4.1. Definimos $SECD$, $DSECD$, $DCUTD$ e $MTZD$ como SEC, $DSEC$, $DCUT$ e MTZ acrescidas da restrição (4.33), respectivamente.

Para a variação do problema **AGD-t**(G, D), as restrições de dependência seriam como em (4.34), apresentada abaixo. Já para **AGD-1**(G, D), elas seriam representadas pelas restrições (4.33) e (4.35), também apresentada abaixo.

$$x_{e'} \geq x_e, \quad \forall e \in E : N^-(e) \neq \emptyset \tag{4.34}$$

$$\sum_{e' \in N^-(e)} x_{e'} \leq 1 + (|N^-(e)| - 1)(1 - x_e), \quad \forall e \in E : N^-(e) \neq \emptyset \tag{4.35}$$

4.3 Desigualdades válidas

Nesta seção, apresentamos algumas desigualdades válidas para o problema. Buscamos com elas melhorar a qualidade da relaxação linear de modelos lineares para **AGDM**(G, D, w), a fim de ter melhores limites inferiores para métodos exatos de resolução.

4.3.1 Desigualdade de dependências nas vizinhanças

Tome **AGDM**(G, D, w) uma instância onde $D(E, A)$ é acíclico. Seja $S = \{e \in E : N^-(e) = \emptyset\}$ o conjunto de fontes de D. Para $e \in E$, seja $h_e = \min\{d(s, e) : s \in S\}$ a distância de e à sua fonte mais próxima. Temos que as seguintes desigualdades são válidas:

$$\sum_{e' \in N_k^-(e) \setminus \{e\}} x_{e'} \geq k x_e, \quad \forall e \in E, \forall k \in [h_e] \tag{4.36}$$

Note que a seleção de x_e implica a seleção de pelo menos h_e outras arestas, pela definição de h_e. Particularmente, k dessas arestas devem ser selecionadas na vizinhança $N_k^-(e)$. Logo, a desigualdade é válida.

Um caso que interessa em nosso estudo ocorre quando D é uma arborescência. Nesse caso, D tem uma única fonte s e é acíclico, havendo um único caminho direcionado entre s e cada outra aresta. Assim, $|N_k^-(e) \setminus \{e\}| = k$, para todos $e \in E$ e $k \in [h_e]$. Logo, as restrições (4.36) são dominadas por $x_{e'} \geq x_e, \forall e \in E, \forall e' \in N_k^-(e), \forall k \in [h_e]$, que são, por sua vez, dominadas pelas restrições de dependência da formulação.

4.3.2 Desigualdade de corte

Tome $G(V, E), D(E, A)$ acíclico e $w : E \to \mathbb{R}^+$ uma instância de **AGDM**(G, D, w). Seja $S \subset E$ um corte em G. Como todo subgrafo conexo de G contém ao menos uma aresta de S, temos que a seguinte desigualdade é válida para **AGDM**(G, D, w):

$$\sum_{e \in S} x_e \geq 1 \tag{4.37}$$

Dentre as, potencialmente em número exponencial, desigualdades do tipo (4.37), julgamos que aquelas com maior impacto na formulação envolvem arestas de G com maior distância em D a partir da fonte mais próxima. Nesse sentido, tomamos S um corte de peso mínimo de G para a função de ponderação $w' : E \to \mathbb{R}^+$, em que $w'(e) = \max\{\frac{1}{d(s,e)} : N^-(s) = \emptyset, s \in N_k^-(e), k \geq 1\}$. Note que $w'(e)$ diminui conforme a distância da fonte mais próxima de e até e aumenta. Com isso, temos que desigualdades (4.37) associadas a cortes mínimos obtidos com essa ponderação asseguram que variáveis cujas arestas encontram-se distantes das fontes mais próximas assumem valores não nulos, assim como algumas variáveis associadas a arestas das quais elas dependem.

Capítulo 5

Testes computacionais

Neste capítulo, apresentamos os experimentos computacionais realizados com base nas formulações matemáticas descritas nos capítulos anteriores. Para tanto, criamos um conjunto de instâncias de **AGDM**(G, D, w). Descrevemos, na seção seguinte, o conjunto de instâncias sobre o qual foram feitos os testes, e nas seções posteriores apreciamos os resultados com eles obtidos.

Adotamos como ambiente de experimentação um mesmo computador constituído de um processador **Intel Core i7 3770** de **3,40 GHz** com **128 KB L1**, **1 MB L2**, **8 MB L3** e **8 threads**, com **16 GB RAM DDR3 1333 MHz**. Em se tratando de software, foram utilizados o sistema operacional **Ubuntu 14.04 LTS**, o software matemático **SageMath 6.5** [17] e o software de otimização **CPLEX 12.4** [18].

5.1 Instâncias

Nesta seção, descrevemos o conjunto de instâncias criadas para a condução dos testes computacionais. Usamos o software **SageMath 6.5** [17] para a geração de grafos e digrafos aleatórios.

Criamos 90 pares $(G(V, E), D(E, A))$ de grafo e digrafo, sendo G conexo e com pesos inteiros atribuídos às suas arestas, escolhidos aleatoriamente entre 1 e 9 (intervalo escolhido arbitrariamente), sendo D uma arborescência. O gerador usa como argumento uma tupla (n, d, b), onde n é o número de vértices de G, d é a probabilidade de existência de uma aresta em G (controlando, pois, a densidade de G, $d(G) = \frac{m(G)}{\binom{n(G)}{2}}$), e b é o grau de saída de cada vértice em D. Esses parâmetros variam, respectivamente, nos conjuntos $N = \{30, 60, 90, 120, 150\}$, $D' = \{0,25; 0,50; 0,75\}$ e $B = \{2, 4, 8\}$. Precisamente, para cada tupla $(n, d, b) \in N \times D' \times B$, criamos dois pares de grafo e digrafo (G, D), com $n(G) = n, d(G) \approx d$

e D uma arborescência onde todos os vértices que não são sumidouros têm grau de saída b, a menos de um único vértice que tem grau de saída $(m(G) - 1) \mod b$. Identificamos o i-ésimo par (G, D) criado conforme a tupla (n, d, b) como sendo a instância (n, d, b, i), para $(n, d, b) \in N \times D' \times B, i \in [2]$.

A construção de D dá-se da seguinte forma: tomamos uma aresta e de E como raiz e definimos D_0 como o digrafo composto apenas pelo vértice isolado e; definimos $E'_0 = E \setminus \{e\}$; na i-ésima iteração, $i \in [k = \lceil \frac{m(G)-1}{b} \rceil]$, selecionamos b arestas de E'_{i-1} - ou as arestas restantes - e as fazemos filhas de um vértice de D_{i-1} que tenha grau de saída zero; definimos E'_i como E'_{i-1} sem as arestas selecionadas e D_i como D_{i-1} alterado como descrito. Tomamos D_k como D.

5.2 Relaxações lineares

Nessa seção, exibimos os resultados das relaxações lineares de dois modelos matemáticos para as instâncias propostas na Seção 5.1. Primeiramente, no entanto, apresentamos algumas considerações sobre os modelos para $\mathbf{AGM}(G, w)$ apresentados na Seção 4.1.

Primeiro, observamos que as relaxações lineares dos modelos SEC, $DSEC$ e $DCUT$ nos fornecem soluções ótimas inteiras para $\mathbf{AGM}(G, w)$, como nos é demonstrado em [9]. Consideramos a princípio, pois, apenas esses três modelos matemáticos.

Como é bastante conhecido da literatura [9], resolve-se as relaxações lineares de SEC, $DSEC$ e $DCUT$ por um processo iterativo em que as restrições do modelo são incluídas à medida em que se verifica que elas estão sendo violadas pela solução ótima atual, aplicando-as como cortes. Tal método é usado devido ao número exponencial, em função de $n(G)$, de restrições que descrevem cada modelo, tornando impraticável sua resolução direta. A separação das restrições de SEC, $DSEC$ e $DCUT$ foi feita segundo proposto em [9].

Usamos o software **CPLEX 12.4** [18] para a resolução das relaxações lineares. Pudemos constatar, com a realização de testes preliminares, que a relaxação linear do modelo $DCUT$, quando acrescida da restrição (4.10) do modelo $DSEC$ e de uma restrição que impede arcos chegando em r de serem selecionados, é a que se resolve aplicando-se o menor número de cortes dentre as relaxações dos modelos SEC, $DSEC$, $DCUT$ e variações consideradas, testados para o mesmo conjunto de instâncias. O modelo que destacamos neste parágrafo é descrito abaixo, e é referenciado como $DCUT2$.

$$\min \quad \sum_{e \in E} w_e x_e \tag{5.1}$$

$$\text{s.a} \quad (x, y) \in P_{DCUT} \tag{5.2}$$

$$\sum_{u \in N(v)} y_{uv} = 1, \qquad \forall v \in V \setminus \{r\} \tag{5.3}$$

$$\sum_{u \in N(r)} y_{ur} = 0 \tag{5.4}$$

$$x \in \mathbb{B}^{m(G)} \tag{5.5}$$

$$y \in \mathbb{B}^{m(D)} \tag{5.6}$$

Avaliamos em seguida MTZ frente a $DCUT2$. Verificamos que a relaxação linear de MTZ fornece valores bem menores que os fornecidos pela de $DCUT2$, porém demanda bem menos tempo computacional que esta última formulação. Com base nessa observação, comparamos, então, as relaxações lineares dos modelos $DCUT2$ e MTZ acrescidos das restrições de dependência, que denotamos por $DCUT2D$ e $MTZD$, respectivamente. Os resultados podem ser vistos na Tabela 5.1, para instâncias com até 90 vértices, e na Tabela 5.2, para aquelas com mais de 90 vértices. Cada linha refere-se a uma instância, que é identificada através da tupla (n, d, b, i). Há uma coluna para o valor ótimo (* significa que ele é desconhecido), seguida de outras três referentes a $DCUT2D$, apresentando, respectivamente, o valor ótimo da relaxação, o gap de integralidade (** significa que não foi possível calculá-lo) e o tempo computacional por ela requerido (em segundos). As três últimas colunas apresentam informações similares, agora com respeito a $MTZD$.

Podemos notar que os valores das relaxações são quase sempre iguais e, quando não são, o valor relaxado de $DCUT2D$ é ligeiramente maior que o de $MTZD$. Essa similaridade entre as relaxações é bastante diferente do que acontece com as formulações base $DCUT2$ e MTZ, cujos valores relaxados podem ser arbitrariamente distantes. Pelo menos para o grupo de instâncias testado, as restrições de dependência foram capazes de fortalecer o modelo baseado nas restrições MTZ a ponto de torná-lo comparável, em termos do limite inferior, àquele baseado em restrições de corte orientado. O gap médio de $MTZD$ para as 90 instâncias é de aproximadamente $9,31\%$, sendo portanto o gap médio de $DCUT2D$ insignificantemente menor que isso.

Podemos ver que a relaxação linear de $MTZD$ é resolvida, com frequência, mais rapidamente que a de $DCUT2D$ para as instâncias com até 90 vértices. A média de tempos,

porém, é muito próxima. Por outro lado, $DCUT2D$ costuma ter tempos de execução menores que os de $MTZD$ para as instâncias com mais de 90 vértices, principalmente nas que apresentam os menores valores b. Para essas instâncias maiores, a média é favorável a $DCUT2D$.

5.3 Modelos inteiros

Nesta seção, comparamos os modelos lineares inteiros $MTZD$ e $DCUT2D$. Antes de tudo, fazemos algumas considerações a respeito de suas implementações.

Como se trata de um modelo simples, a implementação de $MTZD$ não demanda nenhum artifício, mas é importante dizer que sempre consideramos o vértice de índice 0 como raiz e que não usamos os fortalecimentos (4.29), (4.30) e (4.32), pois, apesar de melhorarem minimamente o valor relaxado das instâncias, fizeram com que o *gap* inicial do *branch and bound* - calculado a partir da primeira solução inteira encontrada pelo *solver* e do limite inferior global - aumentasse. Usamos a restrição (4.25) como é apresentada. A implementação de $DCUT2D$, no entanto, envolve o uso de *callbacks*, por se tratar de um modelo cuja resolução requer um método de separação das restrições (4.17) (quanto à inclusão de restrições violadas, incluímos uma por vez, por meio de *LazyConstraintCallback*, sempre que uma nova solução inteira viola alguma restrição). Foi possível resolver $DCUT2D$ apenas serialmente, devido ao uso da *callback*. Já $MTZD$, cuja implementação é simples, pôde ser resolvida usando múltiplas *threads*.

Nas tabelas 5.3 e 5.4, comparamos os tempos de resolução serial e paralela da formulação $MTZD$ (colunas identificadas por $MTZD$-s e $MTZD$, respectivamente) com os de $DCUT2D$. A ocorrência de * indica que o valor ótimo da instância correspondente à linha não foi encontrado durante a resolução dos modelos lineares inteiros. A ocorrência de ** indica que o modelo inteiro correspondente à coluna não foi capaz de encontrar o ótimo para a instância correspondente à linha, devido a memória insuficiente. A ocorrência de *** indica que o modelo inteiro correspondente à coluna não foi capaz de encontrar o ótimo para a instância correspondente à linha em no máximo 500 000 segundos.

Podemos notar que os tempos de execução de $DCUT2D$ são maiores que os da execução serial de $MTZD$ em uma média de aproximadamente $67,35\%$, apesar de $DCUT2D$ ser melhor em alguns poucos casos. Isso mostra que, ao menos para esse conjunto de instâncias, o modelo inteiro $MTZD$ é melhor que $DCUT2D$.

Ainda em relação às tabelas 5.3 e 5.4, podemos notar que a execução paralela do modelo

Tabela 5.1: Resultados das relaxações lineares para instâncias com até 90 vértices.

n	d	b	i	Ótimo	$MTZD$	Gap (%)	Tempo (s)	$DCUT2D$	Gap (%)	Tempo (s)
30	0.25	2	0	138	117,4722	14,87	0,0253	117,4722	14,87	0,0654
30	0.25	2	1	94	93,5000	0,53	0,0291	93,5000	0,53	0,0595
30	0.25	4	0	96	94,0000	2,08	0,0256	94,8333	1,21	0,1810
30	0.25	4	1	82	77,0000	6,09	0,0234	78,0000	4,87	0,1130
30	0.25	8	0	102	98,3333	3,59	0,0248	98,3333	3,59	0,0627
30	0.25	8	1	87	84,0000	3,44	0,0274	84,0000	3,44	0,0584
30	0.50	2	0	142	112,6994	20,63	0,0547	112,6994	20,63	0,1050
30	0.50	2	1	123	107,4628	12,63	0,0609	107,4628	12,63	0,1160
30	0.50	4	0	88	81,7500	7,10	0,0519	81,7500	7,10	0,0962
30	0.50	4	1	80	77,3333	3,33	0,0507	77,3333	3,33	0,0874
30	0.50	8	0	77	74,9166	2,70	0,0468	74,9166	2,70	0,0945
30	0.50	8	1	63	60,7500	3,57	0,0467	60,7500	3,57	0,0858
30	0.75	2	0	119	100,8033	15,29	0,0845	100,8033	15,29	0,1450
30	0.75	2	1	115	96,5847	16,01	0,0855	96,5847	16,01	0,1410
30	0.75	4	0	70	66,2500	5,35	0,0793	66,2500	5,35	0,1300
30	0.75	4	1	90	79,1927	12,00	0,0719	79,1927	12,00	0,1330
30	0.75	8	0	69	66,2000	4,05	0,0736	66,2000	4,05	0,1310
30	0.75	8	1	60	59,8333	0,27	0,0700	59,8333	0,27	0,1120
60	0.25	2	0	202	182,5695	9,61	0,1450	182,5695	9,61	0,2710
60	0.25	2	1	237	215,7922	8,94	0,1530	215,7922	8,94	0,3100
60	0.25	4	0	163	158,2500	2,91	0,1210	158,2500	2,91	0,2190
60	0.25	4	1	166	159,5238	3,90	0,1190	159,5238	3,90	0,2630
60	0.25	8	0	154	148,4722	3,58	0,1120	148,4722	3,58	0,2330
60	0.25	8	1	156	147,0500	5,73	0,0978	147,0500	5,73	0,2230
60	0.50	2	0	226	196,7378	12,94	0,4410	196,7378	12,94	0,5070
60	0.50	2	1	228	192,8107	15,43	0,4670	192,8107	15,43	0,5060
60	0.50	4	0	154	148,9206	3,29	0,2670	148,9206	3,29	0,4110
60	0.50	4	1	168	152,7824	9,05	0,3210	153,0318	8,90	0,5930
60	0.50	8	0	123	119,9017	2,51	0,2400	119,9068	2,51	0,5330
60	0.50	8	1	164	146,8349	10,46	0,2780	146,8349	10,46	0,4040
60	0.75	2	0	189	165,1549	12,61	0,7420	165,1549	12,61	0,7330
60	0.75	2	1	203	170,3927	16,06	0,6740	170,3927	16,06	0,7420
60	0.75	4	0	144	129,9676	9,74	0,7260	129,9676	9,74	0,6200
60	0.75	4	1	141	130,3855	7,52	0,6690	130,3855	7,52	0,6050
60	0.75	8	0	121	108,4472	10,37	0,7100	108,4472	10,37	0,5990
60	0.75	8	1	124	112,8258	9,01	0,6740	112,8258	9,01	0,5800
90	0.25	2	0	342	275,9718	19,30	0,6930	275.9718	19,30	0,7370
90	0.25	2	1	349	294,8489	15,51	0,6810	294,8489	15,51	0,8250
90	0.25	4	0	251	227,3855	9,40	0,6340	227,3855	9,40	0,6800
90	0.25	4	1	243	228,6917	5,88	0,6650	228,6917	5,88	0,6400
90	0.25	8	0	198	189,6006	4,24	0,3480	189,6006	4,24	0,7090
90	0.25	8	1	210	195,1674	7,06	0,3920	195,2324	7,03	1,0300
90	0.50	2	0	338	285,5183	15,52	1,8700	285,5183	15,52	1,4200
90	0.50	2	1	320	270,2254	15,55	2,0100	270,2254	15,55	1,4200
90	0.50	4	0	237	212,1950	10,46	1,1100	212,1950	10,46	1,2300
90	0.50	4	1	222	206,6716	6,90	1,0000	206,6716	6,90	1,2000
90	0.50	8	0	201	189,0746	5,93	1,9300	189,0746	5,93	1,1600
90	0.50	8	1	193	177,7894	7,88	1,6600	177,7894	7,88	1,0800
90	0.75	2	0	316	266,0699	15,80	3,8800	266,0699	15,80	2,1500
90	0.75	2	1	320	253,1779	20,88	4,5200	253,1779	20,88	2,3300
90	0.75	4	0	212	187,9366	11,35	1,8900	187,9366	11,35	1,9300
90	0.75	4	1	218	197,4896	9,40	0,1980	197,4896	9,40	1,8500
90	0.75	8	0	183	169,0398	7,62	1,6600	169,0398	7,62	1,6800
90	0.75	8	1	169	156,8259	7,20	1,6300	156,8259	7,20	1,7600
				Média		8,95	0,6418		8,90	0,6314

Tabela 5.2: Resultados das relaxações lineares para instâncias com mais de 90 vértices.

n	d	b	i	Ótimo	$MTZD$	Gap (%)	Tempo (s)	$DCUT2D$	Gap (%)	Tempo (s)
120	0.25	2	0	434	348,3422	19,73	1,77	348,3422	19,73	1,630
120	0.25	2	1	466	380,7706	18,28	1,76	380,7706	18,28	1,560
120	0.25	4	0	300	288,4154	3,86	1,59	288,4154	3,86	1,210
120	0.25	4	1	321	299,4554	6,71	1,57	299,4554	6,71	0,127
120	0.25	8	0	233	226,7117	2,69	1,47	226,9778	2,58	1,740
120	0.25	8	1	246	237,1638	3,59	1,64	237,1638	3,59	1,470
120	0.50	2	0	398	331,1002	16,80	5,81	331,1002	16,80	3,300
120	0.50	2	1	420	343,5908	18,19	5,34	343,5908	18,19	3,210
120	0.50	4	0	302	274,1688	9,21	3,10	274,1688	9,21	2,620
120	0.50	4	1	295	280,7848	4,81	4,98	280,7848	4,81	2,680
120	0.50	8	0	231	219,9000	4,80	2,20	219,9000	4,80	2,380
120	0.50	8	1	254	232,0141	8,65	2.33	232,0141	8,65	2,410
120	0.75	2	0	403	334,2481	17,06	12,30	334,2481	17,06	5,820
120	0.75	2	1	409	361,9028	11,51	12,90	361,9028	11,51	5,560
120	0.75	4	0	289	261,9424	9,36	5,51	261,9424	9,36	4,450
120	0.75	4	1	300	270,3159	9,89	5,99	270,3159	9,89	4,240
120	0.75	8	0	212	202,5500	4,45	4,62	202,5500	4,45	3,760
120	0.75	8	1	241	222,2533	7,77	4,87	222,2533	7,77	3,940
150	0.25	2	0	538	459,4907	14,59	4,85	459,4907	14,59	3,230
150	0.25	2	1	536	453,0966	15,46	4,46	453,0966	15,46	3,100
150	0.25	4	0	355	332,3276	6,38	3,91	332,4342	6,35	3,110
150	0.25	4	1	383	369,5023	3,52	4,01	369,5023	3,52	2,510
150	0.25	8	0	321	308,7039	3,83	2,07	308,7039	3,83	2,810
150	0.25	8	1	321	295,1328	8,05	1,96	295,1328	8,05	2,220
150	0.50	2	0	495	418,7787	15,39	9,95	418,7787	15,39	6,650
150	0.50	2	1	546	446,3734	18,24	15,10	446,3734	18,24	7,000
150	0.50	4	0	362	324,8102	10,27	6,74	324,8102	10,27	5,290
150	0.50	4	1	359	336,1449	6,36	13,40	336,1449	6,36	5,150
150	0.50	8	0	294	265,9665	9,53	6,13	265,9665	9,53	4,780
150	0.50	8	1	309	285,9431	7,46	6,21	285,9431	7,46	4,750
150	0.75	2	0	483	399,5253	17,28	26,80	399,5253	17,28	11,700
150	0.75	2	1	*	412,0502	**	28,80	412,0502	**	12,500
150	0.75	4	0	363	318,9587	12,13	12,50	318,9587	12,13	8,630
150	0.75	4	1	376	327,9563	12,77	12,50	327,9563	12,77	9,050
150	0.75	8	0	273	252,0400	7,67	11,20	252,0400	7,67	7,920
150	0.75	8	1	273	249,9727	8,43	10,80	249,9727	8,43	8,000
				Média		10,14	7,56		10,13	4,458

inteiro $MTZD$ proporciona um *speedup* médio aproximado de apenas 1,68, apesar das oito *threads* utilizadas. Alertamos, porém, ser grosseiro o cálculo do *speedup* com base nos tempos de execução apresentados, visto que, como o **CPLEX** é um *software* de código fechado, não podemos saber se a execução serial provoca alguma outra alteração no modo como ele resolve os modelos.

Com os dados da Tabela 5.5, que apresenta os tempos de execução médios por densidade na coluna TEM, notamos que o problema toma mais tempo de execução para ser resolvido à medida que o grafo G das instâncias testadas tem maior densidade. Fazendo uso da Tabela 5.6, que exibe os tempos de execução médios por ramificação na coluna TEM, percebemos que o problema toma menos tempo de execução para ser resolvido ao passo que a arborescência D das instâncias testadas tem maior grau de ramificação.

5.4 Desigualdade válida

Nesta seção, tratamos do tempo de execução do modelo (inteiro) $MTZD$, quando a ele acrescida a desigualdade válida proposta na Subseção 4.3.2. Denotamos tal modelo por $MTZD\text{-}CUT$. A outra desigualdade válida, proposta na Subseção 4.3.1, é inóqua quando D é arborescência e, portanto, não foi usada em nossos testes.

Levamos em conta, na Tabela 5.7, apenas as instâncias cujos valores ótimos são conhecidos e cujo tempo de execução no modelo $MTZD$ é de pelo menos 100 segundos. Das 90, 20 instâncias atingem esse requisito. A instância $(150, 0.50, 2, 1)$, no entanto, é a que toma mais tempo de resolução dentre todas as resolvidas, e portanto também não é levada em conta agora.

Apesar de $MTZD\text{-}CUT$ apresentar uma melhoria significativa para algumas instâncias em relação a $MTZD$, existem casos onde ocorre uma grande piora no tempo de execução. Temos, com isso, uma média de melhoria aproximada de apenas 7,31%, muito próxima da mediana de melhoria, também aproximada, de 7,24%.

5.5 Solução inicial

Nesta seção, descrevemos uma restrição no espaço viável de $\mathbf{AGDM}(G, D, w)$ que contém apenas soluções viáveis contidas em um certo subgrafo gerador de G. Tal subgrafo é selecionado levando em conta as restrições de dependência descritas por D e os pesos w.

Seja $G(V, E), D(E, A)$ e $w : E \to \mathbb{R}^+$ a entrada de $\mathbf{AGDM}(G, D, w)$. Vamos definir uma ordem para as arestas em E com base na função $f : E \to \mathbb{R}^+$, definida por $f(e) =$

Tabela 5.3: Tempos de execução dos modelos inteiros para instâncias com até 90 vértices.

n	d	b	i	Ótimo	$MTZD$ (s)	$MTZD$-s (s)	$DCUT2D$ (s)
30	0.25	2	0	138	0,01	0,01	0,03
30	0.25	2	1	94	0,01	0,01	0,02
30	0.25	4	0	96	0,02	0,02	0,02
30	0.25	4	1	82	0,02	0,02	0,02
30	0.25	8	0	102	0,01	0,01	0,01
30	0.25	8	1	87	0,01	0,01	0,03
30	0.50	2	0	142	0,01	0,01	0,02
30	0.50	2	1	123	0,11	0,19	0,29
30	0.50	4	0	88	0,02	0,03	0,05
30	0.50	4	1	80	0,02	0,01	0,03
30	0.50	8	0	77	0,02	0,01	0,03
30	0.50	8	1	63	0,01	0,01	0,01
30	0.75	2	0	119	0,09	0,25	0,59
30	0.75	2	1	115	0,22	0,41	0,59
30	0.75	4	0	70	0,03	0,03	0,02
30	0.75	4	1	90	0,09	0,20	0,22
30	0.75	8	0	69	0,02	0,02	0,04
30	0.75	8	1	60	0,01	0,01	0,02
60	0.25	2	0	202	0,48	0,47	0,82
60	0.25	2	1	237	0,59	0,78	1,66
60	0.25	4	0	163	0,14	0,09	0,11
60	0.25	4	1	166	0,06	0,06	0,09
60	0.25	8	0	154	0,04	0,04	0,08
60	0.25	8	1	156	0,10	0,09	0,07
60	0.50	2	0	226	1,48	3,26	5,17
60	0.50	2	1	228	2,27	3,77	7,60
60	0.50	4	0	154	0,23	0,21	0,33
60	0.50	4	1	168	0,89	1,06	1,68
60	0.50	8	0	123	0,21	0,39	0,47
60	0.50	8	1	164	0,69	0,81	1,20
60	0.75	2	0	189	1,97	2,37	4,75
60	0.75	2	1	203	1,68	2,52	8,49
60	0.75	4	0	144	1,12	1,10	2,58
60	0.75	4	1	141	0,27	0,40	0,64
60	0.75	8	0	121	0,50	0,67	1,05
60	0.75	8	1	124	0,26	0,24	0,48
90	0.25	2	0	342	3,61	8,82	10,02
90	0.25	2	1	349	10,87	30,45	104,16
90	0.25	4	0	251	1,59	2,31	2,79
90	0.25	4	1	243	1,03	0,63	0,89
90	0.25	8	0	198	0,12	0,11	0,15
90	0.25	8	1	210	0,42	0,37	0,68
90	0.50	2	0	338	216,35	478,86	256,85
90	0.50	2	1	320	36,50	333,65	158,36
90	0.50	4	0	237	3,72	6,19	7,78
90	0.50	4	1	222	4,85	7,17	6,64
90	0.50	8	0	201	1,14	1,31	1,73
90	0.50	8	1	193	1,61	2,17	3,83
90	0.75	2	0	316	107,51	227,79	584,20
90	0.75	2	1	320	39,34	408,48	208,99
90	0.75	4	0	212	21,78	44,03	50,27
90	0.75	4	1	218	20,04	84,50	43,03
90	0.75	8	0	183	2,94	2,87	3,41
90	0.75	8	1	169	3,68	2,78	5,46
				Média	9,08	30,77	27,56

Tabela 5.4: Tempos de execução dos modelos inteiros para instâncias com mais de 90 vértices.

n	d	b	i	Ótimo	$MTZD$ (s)	$MTZD$-s (s)	$DCUT2D$ (s)
120	0.25	2	0	434	683,87	1476,26	2359,79
120	0.25	2	1	466	600,08	3317,28	8020,24
120	0.25	4	0	300	2,06	1,73	4,48
120	0.25	4	1	321	4,85	5,94	4,11
120	0.25	8	0	233	0,60	0,50	1,00
120	0.25	8	1	246	0,76	0,63	0,93
120	0.50	2	0	398	3489,66	6613,45	6802,91
120	0.50	2	1	420	2923,59	2470,64	8719,21
120	0.50	4	0	302	141,83	194,11	570,71
120	0.50	4	1	295	11,86	13,50	12,99
120	0.50	8	0	231	3,66	2,97	4,97
120	0.50	8	1	254	10,55	12,82	17,27
120	0.75	2	0	403	4960,16	5090,58	16322,59
120	0.75	2	1	409	1958,91	3899,00	6163,77
120	0.75	4	0	289	982,21	1391,13	2379,59
120	0.75	4	1	300	206,16	863,92	292,57
120	0.75	8	0	212	4,61	5,85	4,85
120	0.75	8	1	241	19,77	13,60	32,83
150	0.25	2	0	538	1305,78	1677,46	6352,16
150	0.25	2	1	536	2263,90	6219,27	42304,22
150	0.25	4	0	355	9,33	14,10	11,27
150	0.25	4	1	383	7,20	6,75	7,76
150	0.25	8	0	321	11,69	14,01	9,44
150	0.25	8	1	321	2,93	2,50	2,56
150	0.50	2	0	495	4286,03	6250,46	10614,8
150	0.50	2	1	546	63243,97	111057,72	412665,05
150	0.50	4	0	362	2462,97	12326,15	5108,39
150	0.50	4	1	359	196,81	213,07	165,97
150	0.50	8	0	294	44,18	54,66	55,28
150	0.50	8	1	309	62,46	65,28	47,83
150	0.75	2	0	483	32807,35	96188,49	84107,06
150	0.75	2	1	*	**	**	***
150	0.75	4	0	363	4308,08	8943,04	6955,49
150	0.75	4	1	376	11197,05	7757,59	32463,48
150	0.75	8	0	273	41,45	77,68	79,91
150	0.75	8	1	273	49,55	56,74	105,44
				Média	2207,70	4860,03	7061,93

Tabela 5.5: Tempos de execução médios por densidade.

n	d	TEM
30	0.25	0,01
30	0.50	0,03
30	0.75	0,07
60	0.25	0,23
60	0.50	0,96
60	0.75	0,96
90	0.25	2,94
90	0.50	44,02
90	0.75	32,54
120	0.25	215,37
120	0.50	1096,85
120	0.75	1355,30
150	0.25	600,13
150	0.50	1410,49
150	0.75	9680,69

Tabela 5.6: Tempos de execução médios por ramificação.

n	b	TEM
30	2	0,07
30	4	0,03
30	8	0,01
60	2	1,41
60	4	0,45
60	8	0,30
90	2	69,03
90	4	8,83
90	8	1,65
120	2	2436,04
120	4	224,82
120	8	6,65
150	2	10165,76
150	4	3030,24
150	8	35,37

$w_e + \min\{f(e') : e' \in N^-(e)\}$. Note que se $e \in E$ for uma fonte em D, teremos $f(e) = w_e$. Note ainda que f atribui a $e \in E$ o peso do caminho direcionado de menor custo de uma fonte de D para e.

Considere $E' = E \setminus S$, onde S são as fontes de D. Considere também uma ordenação para $E' = \{e_1, e_2, \ldots, e_{m(G)-|S|}\}$ em que $i \leq j \implies f(e_i) \leq f(e_j)$. Para $k \in [m(G) - |S|]$, seja E'_k o subconjunto de E' formado pelas primeiras $\frac{km(G)}{\log_2(n(G))}$ arestas de E'. Considere $G_k(V, S \cup E'_k)$ o subgrafo gerador de G que contém apenas as arestas em $S \cup E'_k$, assim como $D_k(S \cup E'_k, A(S \cup E'_k))$ o digrafo induzido pelas mesmas, para $k \in \mathbb{N}$. Definimos P_k como o subproblema $\mathbf{AGDM}(G_k, D_k, w), k \in \mathbb{N}$.

A heurística que propomos nesta seção consiste em resolver sucessivamente, para $k \geq 3$, os problemas P_k, até o primeiro deles se tornar viável. Assim, a solução ótima desse problema fornece uma solução inicial para $\mathbf{AGDM}(G, D, w)$, que é usada também para fornecer um limite superior para seu valor ótimo. Aqui, escolhemos resolver os problemas $P_k, k \geq 3$, usando o modelo inteiro $MTZD$.

Na Tabela 5.8, apresentamos os resultados computacionais da heurística proposta, para as instâncias cujos valores ótimos são conhecidos e cujo tempo de execução no modelo $MTZD$ é de pelo menos 100 segundos. Novamente desconsideramos $(150, 0.50, 2, 1)$ e $(150, 0.75, 2, 1)$. Podemos verificar que a média aproximada do *gap* percentual é de 1,59%, com mediana aproximada de 0,66%. Notamos também que o tempo de execução da heurística

Tabela 5.7: Tempos de execução com a desigualdade válida.

n	d	b	i	$MTZD$ (s)	$MTZD$-CUT (s)	Melhoria (%)
90	0.50	2	0	216,35	44,53	79,41
90	0.75	2	0	107,51	107,56	-0,04
120	0.25	2	0	683,87	775,57	-13,40
120	0.25	2	1	600,08	673,25	-12,19
120	0.50	2	0	3489,66	3778,74	-8,28
120	0.50	2	1	2923,59	1625,17	44,41
120	0.50	4	0	141,83	95,94	32,35
120	0.75	2	0	4960,16	5639,58	-13,69
120	0.75	2	1	1958,91	2144,94	-9,49
120	0.75	4	0	982,21	379,19	61,39
120	0.75	4	1	206,16	140,45	31,87
150	0.25	2	0	1305,78	1313,03	-0,55
150	0.25	2	1	2263,90	3325,06	-46,87
150	0.50	2	0	4286,03	2815,97	34,29
150	0.50	4	0	2462,97	6773,74	-175,02
150	0.50	4	1	196,81	102,39	47,97
150	0.75	2	0	32807,35	30429,16	7,24
150	0.75	4	0	4308,08	3397,7	21,13
150	0.75	4	1	11197,05	4661,6	58,36
Média				3952,54	3590,71	7,31
Mediana				1958,91	1625,17	7,24

proporciona uma economia média de 92,98% em relação ao tempo de $MTZD$, com mediana de 94,70%.

5.6 $MTZD$ com solução inicial

Nesta seção, apresentamos os tempos de execução obtidos ao resolver o modelo $MTZD$ partindo da solução inicial obtida com a heurística descrita na Seção 5.5, para as instâncias cujos valores ótimos são conhecidos e cujo tempo de execução no modelo $MTZD$ é de pelo menos 100 segundos, novamente desconsiderando $(150, 0.50, 2, 1)$ e $(150, 0.75, 2, 1)$. Apresentamos, na Tabela 5.9, os tempos de execução de $MTZD$ quando partindo da citada solução inicial, na coluna $MTZD$-start, relacionados com os tempos de execução de $MTZD$.

Podemos notar resultados muito variados, com ocorrência de melhorias e pioras nos tempos $MTZD$-start em relação a $MTZD$. Percebemos, inclusive, uma média aproximada de piora de $21,90\%$, contrapondo-se a uma mediana aproximada de melhoria de $11,78\%$. A mediana positiva indica que houve piora em menos de metade das instâncias testadas. A média negativa indica, no entanto, que tal piora foi muito mais significativa que a melhoria

Tabela 5.8: Resultados da heurística de restrição do espaço de busca.

n	d	b	i	Ótimo	$MTZD$ (s)	Heurística	Tempo (s)	Gap (%)	Melhoria (%)
90	0.50	2	0	338	216,35	339	4,78	0,29	97,79
90	0.75	2	0	316	107,51	316	14,35	0,00	86,65
120	0.25	2	0	434	683,87	436	9,61	0,46	98,59
120	0.25	2	1	466	600,08	494	5,17	6,00	99,13
120	0.50	2	0	398	3489,66	407	387,18	2,26	88,90
120	0.50	2	1	420	2923,59	443	251,77	5,47	91,38
120	0.50	4	0	302	141,83	305	12,63	0,99	91,09
120	0.75	2	0	403	4960,16	403	100,91	0,00	97,96
120	0.75	2	1	409	1958,91	409	103,68	0,00	94,70
120	0.75	4	0	289	982,21	289	43,23	0,00	95,59
120	0.75	4	1	300	206,16	302	24,45	0,66	88,14
150	0.25	2	0	538	1305,78	554	5,66	2,97	99,56
150	0.25	2	1	536	2263,90	575	8,11	7,27	99,64
150	0.50	2	0	495	4286,03	499	54,43	0,80	98,73
150	0.50	4	0	362	2462,97	365	515,72	0,82	79,06
150	0.50	4	1	359	196,81	361	18,11	0,55	90,79
150	0.75	2	0	483	32807,35	483	3152,00	0,00	90,39
150	0.75	4	0	363	4308,08	368	221,3	1,37	94,86
150	0.75	4	1	376	11197,05	377	1828,63	0,26	83,66
				Média	3952,54		355,88	1,59	92,98
				Mediana	1958,91		43,23	0,66	94,70

apresentada.

5.7 $MTZD$ com solução inicial e prioridades

Nesta seção, apresentamos uma proposta de priorizar a ramificação de algumas variáveis do modelo $MTZD$. Como mostrou melhoria na maioria das instâncias consideradas, vamos partir da técnica apresentada na Seção 5.6.

Dados $(G(V,E), D(E,A))$, com D acíclico, a partir de uma instância de **AGDM**(G, D, w), sejam $S \subseteq E$ o conjunto de fontes de D e $\epsilon = \min\{\epsilon(s) : s \in S\}$ a menor excentricidade de uma fonte. Definimos o conjunto $E' \subset E$ como $E' = \{e \in E : \min\{d(s,e) : s \in S\} = \lfloor \frac{\epsilon}{3} \rfloor\}$. Pensamos em E' como o conjunto de arestas que estão a uma distância intermediária de sua fonte mais próxima. Vamos priorizar a ramificação das variáveis $x_e, e \in E'$, pois os subproblemas originados dessa ramificação apresentam a seguinte tendência:

- quando $x_e = 1$, temos um ganho no limite inferior por e manter-se a uma certa distância da fonte mais próxima, o que força a fixação em 1 de um número considerável de outras

Tabela 5.9: Tempos de execução de $MTZD$ com solução inicial.

n	d	b	i	$MTZD$ (s)	$MTZD$-start (s)	Melhoria (%)
90	0.50	2	0	216,35	55,43	74,37
90	0.75	2	0	107,51	88,90	17,31
120	0.25	2	0	683,87	1225,65	-79,22
120	0.25	2	1	600,08	775,02	-29,15
120	0.50	2	0	3489,66	8603,43	-146,54
120	0.50	2	1	2923,59	2037,87	30,29
120	0.50	4	0	141,83	107,14	24,45
120	0.75	2	0	4960,16	4429,87	10,69
120	0.75	2	1	1958,91	502,70	74,33
120	0.75	4	0	982,21	315,61	67,86
120	0.75	4	1	206,16	181,86	11,78
150	0.25	2	0	1305,78	290,68	77,73
150	0.25	2	1	2263,90	3369,85	-48,85
150	0.50	2	0	4286,03	3321,23	22,51
150	0.50	4	0	2462,97	10498,11	-326,23
150	0.50	4	1	196,81	104,78	46,76
150	0.75	2	0	32807,35	61092,21	-86,21
150	0.75	4	0	4308,08	4206,62	2,35
150	0.75	4	1	11197,05	29159,87	-160,42
Média				3952,54	6861,41	-21,90
Mediana				1958,91	1225,65	11,78

variáveis x pelas restrições de dependência.

- quando $x_e = 0$, temos que as variáveis associadas às arestas que dependem, mesmo que transitivamente, de e assumem valor zero, e como e não se encontra tão distante da fonte mais próxima, essas costumam ser muitas variáveis.

É preciso esclarecer que o valor $\lfloor \frac{\epsilon}{3} \rfloor$ foi concebido empiricamente. Seu intuito é acentuar o efeito proporcionado pela ramificação das variáveis $x_e, e \in E'$. Esclarecemos também que a ramificação priorizada das variáveis em questão é implementada aumentando sua prioridade no **CPLEX**.

Na Tabela 5.10, apresentamos os tempos de execução, para instâncias cujos valores ótimos são conhecidos e cujo tempo de execução no modelo $MTZD$ é de pelo menos 100 segundos, do modelo $MTZD$ iniciando com a solução obtida pelo método descrito na Seção 5.5 e priorizando a ramificação de variáveis como descrito no parágrafo anterior. Aqui também desconsideramos as instâncias $(150, 0.50, 2, 1)$ e $(150, 0.75, 2, 1)$. Podemos notar, pela Tabela 5.10, uma melhoria significativa em várias instâncias, acompanhada de uma certa piora em algumas outras. Obtemos uma média aproximada de melhoria de 29,69%, acompanhada de

uma mediana aproximada de 39,15% no tempo de execução.

Tabela 5.10: Tempos de execução de $MTZD$ com solução inicial e variáveis priorizadas.

n	d	b	i	$MTZD$ (s)	$MTZD$-start-priorities (s)	Melhoria (%)
90	0.50	2	0	216,35	31,17	85,59
90	0.75	2	0	107,51	91,10	15,26
120	0.25	2	0	683,87	703,03	-2,80
120	0.25	2	1	600,08	353,40	41,10
120	0.50	2	0	3489,66	3191,55	8,54
120	0.50	2	1	2923,59	1422,61	51,34
120	0.50	4	0	141,83	84,82	40,19
120	0.75	2	0	4960,16	2610,95	47,36
120	0.75	2	1	1958,91	792,18	59,56
120	0.75	4	0	982,21	209,14	78,70
120	0.75	4	1	206,16	171,69	16,72
150	0.25	2	0	1305,78	173,18	86,73
150	0.25	2	1	2263,90	2606,37	-15,12
150	0.50	2	0	4286,03	6470,05	-50,95
150	0.50	4	0	2462,97	2465,26	-0,09
150	0.50	4	1	196,81	82,58	58,04
150	0.75	2	0	32807,35	42337,39	-29,04
150	0.75	4	0	4308,08	2621,25	39,15
150	0.75	4	1	11197,05	7406,94	33,84
Média				3952,54	3885,50	29,69
Mediana				1958,91	792,18	39,15

5.8 Instâncias difíceis

Tratamos, nesta seção, das instâncias $(150, 0.50, 2, 1)$ e $(150, 0.75, 2, 1)$. Essas foram separadas das demais por serem, respectivamente, a instância que tomou mais tempo de execução no modelo $MTZD$ e a única instância cujo ótimo é desconhecido. Lembramos que, na Seção 5.3, $(150, 0.50, 2, 1)$ tem seu valor ótimo 546 encontrado em 63243,97 segundos pelo modelo $MTZD$.

Com a heurística apresentada na Seção 5.5, pudemos encontrar, para a instância $(150, 0.50, 2, 1)$, uma solução viável de custo 551 em 2259,08 segundos, o que resulta em um gap de 0,91% e uma melhoria de 96,42 % no tempo de execução. Já para a instância $(150, 0.75, 2, 1)$, cujo valor ótimo é desconhecido, a mesma heurística foi capaz de encontrar uma solução viável de custo 522 em 18215,80 segundos.

Usando a técnica de resolução apresentada na Seção 5.7, resolvemos a instância $(150, 0.50, 2, 1)$ em 73576,92 segundos, o que representa uma piora de 16,33% em relação

ao tempo de $MTZD$. Já a instância $(150, 0.75, 2, 1)$ não pôde ser resolvida pela mesma técnica, devido a memória insuficiente.

Capítulo 6

Conclusão

Neste capítulo, finalizamos o presente trabalho apreciando os resultados teóricos e computacionais alcançados com as reduções, técnicas e modelos matemáticos propostos. Expomos também nossas pretensões quanto a trabalhos futuros para o estudo do **AGDM**(G, D, w) e de seus problemas relacionados.

6.1 Considerações

Exibimos agora nossas conclusões sobre os resultados obtidos. Consideramos primeiro a parte teórica para em seguida tratarmos dos testes computacionais.

procurando marcar a fronteira entre os casos fáceis e difíceis do problema, identificamos estruturas simples para o digrafo D que possibilitam solução em tempo polinomial

Com as proposições estabelecidas no Capítulo 3, podemos notar que **AGDM**(G, D, w) é de difícil resolução mesmo quando G e D apresentam estruturas simples. Dessa maneira, consideramos que os resultados de **NP-completude** obtidos para o problema são fortes. Por outro lado, a redução proposta na Subseção 3.4.2 teve por objetivo mostrar que **AGDM**(G, D, w) permanece difícil mesmo se impusermos que as restrições de dependência ocorram apenas entre arestas adjacentes, um requisito que consideramos realista para futuras aplicações práticas. Adicionalmente, identificamos certas estruturas para o digrafo D que facilitam a resolução do problema. Ainda no Capítulo 3, relacionamos o problema definido neste trabalho com outros, especialmente derivando reduções desses para aquele, de modo a certificar sua abrangência.

No Capítulo 4, apresentamos modelos de programação inteira para **AGDM**(G, D, w) diretamente derivados de formulações para árvore geradora mínima. No tocante às relaxações lineares, podemos notar algo bastante peculiar entre os modelos MTZ e $DCUT2$ e seus

relacionados, $MTZD$ e $DCUT2D$. Enquanto a diferença absoluta entre os valores ótimos das relaxações lineares de MTZ e $DCUT2$ é conhecidamente significativa, o mesmo não se aplica a $MTZD$ e $DCUT2D$. Entre esses últimos, ao contrário, a diferença mostra-se ínfima, e em muitos casos o valor de suas relaxações chega a ser igual. Isso sugere que as restrições de dependência têm muita influência no modelo, sendo decisivas na definição do politopo do problema, ao menos para o conjunto de instâncias testadas.

Também no Capítulo 4 são apresentadas algumas desigualdades válidas, dentre as quais apenas uma parecia promissora para nossas instâncias de teste. Mesmo assim, ela apresentou um ganho de desempenho médio muito baixo, e não foi considerada nos testes seguintes.

No Capítulo 5, verificamos a influência que a densidade de G e o grau de ramificação de D, caso esse seja uma arborescência, têm sobre a dificuldade de se resolver $\mathbf{AGDM}(G, D, w)$. Percebemos que ocorre uma explosão combinatorial em $\mathbf{AGDM}(G, D, w)$ quando G tem entre 90 e 120 vértices.

Ainda no Capítulo 5, propusemos um método heurístico capaz de chegar bem próximo do valor ótimo das instâncias testadas, levando bem menos tempo que sua resolução exata. Constatamos, no entanto, que ao usar a solução obtida por esse método como ponto inicial para a resolução exata das instâncias, há tanto melhoria como piora nos tempos de execução, sendo a piora mais significativa, embora menos frequente.

Continuando a exposição do Capítulo 5, propusemos ainda, com base na estrutura de D, um conjunto de variáveis cuja ramificação deve ser priorizada. Testamos essa técnica sobre o modelo $MTZD$, partindo da solução encontrada com o método heurístico proposto. Isso nos trouxe resultados interessantes. Apesar de haver piora em algumas instâncias, obtivemos boas porcentagens de melhoria média.

6.2 Trabalhos futuros

Nesta seção, tratamos das nossas intenções quanto ao futuro estudo de $\mathbf{AGDM}(G, D, w)$ e seus problemas relacionados. Como na seção anterior, fazemos primeiro considerações no âmbito teórico, para em seguida apresentarmos propostas de métodos de resolução que podem ser desenvolvidas em algum momento.

O presente trabalho se encerra deixando de analisar a complexidade de $\mathbf{AGDM}(G, D, w)$ para algumas classes de grafos simples, sobre as quais temos a suspeita de haver algoritmos eficientes. É o que ocorre, por exemplo, quando D é a orientação de um emparelhamento.

Temos ainda dois problemas relacionados a explorar: quando se pede que todas as de-

pendências de uma aresta ou exatamente uma delas seja(m) escolhida(s) para possibilitar a escolha dessa aresta. É visível, entretanto, que **AGDM**(G, D, w) possui restrições de dependência mais leves que suas variantes, embora isso não seja motivo para desprezá-las.

As restrições de dependência em si mostraram-se um tanto fortes, ao fazer $MTZD$ ter um valor relaxado muito similar ao de $DCUT2D$. Pode haver interesse em verificar a complexidade de problemas clássicos quando a eles acrescentamos restrições análogas às aqui estudadas.

No tocante ao aspecto prático do problema, temos que o *gap* entre o valor ótimo relaxado de $MTZD$ e seu valor ótimo inteiro ainda é explorável. Isso nos inspira a buscar mais desigualdades válidas para fortalecer o valor relaxado desse modelo. Note que aqui encontramos um modo de fortalecê-lo durante a resolução, e que tivemos pouco sucesso ao fazê-lo por aplicações de desigualdades antes de resolver o modelo.

Referências Bibliográficas

[1] AKGÜN, İbrahim; TANSEL, Barbaros. New formulations of the hop-constrained minimum spanning tree problem via Miller–Tucker–Zemlin constraints. *European Journal of Operational Research*, v. 212, n. 2, p. 263-276, 2011.

[2] DARMANN, Andreas et al. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, v. 159, n. 16, p. 1726-1735, 2011.

[3] DE ALMEIDA, Ana; MARTINS, Pedro; DE SOUZA, Maurício. Min-degree constrained minimum spanning tree problem: complexity, properties, and formulations. *International Transactions in Operational Research*, v. 19, n. 3, p. 323-352, 2012.

[4] DESROCHERS, Martin; LAPORTE, Gilbert. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, v. 10, n. 1, p. 27-36, 1991.

[5] FREDMAN, Michael; TARJAN, Robert. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, v. 34, n. 3, p. 596-615, 1987.

[6] GAREY, Michael; JOHNSON, David. Computers and intractability: a guide to the theory of NP-completeness. 1979. San Francisco, LA: Freeman, 1979.

[7] KARA, Imdat. Tightening bounding constraints of the Miller-Tucker-Zemlin based formulation of the capacitated vehicle routing problems and some extensions. *Proceeding of the 2nd International Conference on Manufacturing Engineering, Quality and Production Systems* (Eds. C. Panait et al.), WSEAS Press, Constantza, Romania, p. 137-142, 2010.

[8] KRUSKAL, Joseph. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, v. 7, n. 1, p. 48-50, 1956.

[9] MAGNANTI, Thomas; WOLSEY, Laurence. Optimal trees. *Handbooks in operations research and management science*, v. 7, p. 503-615, 1995.

[10] MILLER, Clair; TUCKER, Albert; ZEMLIN, Richard. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, v. 7, n. 4, p. 326-329, 1960.

[11] NARULA, Subhash; HO, Cesar. Degree-constrained minimum spanning tree. *Computers & Operations Research*, v. 7, n. 4, p. 239-249, 1980.

[12] PRIM, Robert. Shortest connection networks and some generalizations. *Bell system technical journal*, v. 36, n. 6, p. 1389-1401, 1957.

[13] SAMER, Phillippe; URRUTIA, Sebastián. A branch and cut algorithm for minimum spanning trees under conflict constraints. *Optimization Letters*, v. 9, n. 1, p. 41-55, 2015.

[14] SHERALI, Hanif; DRISCOLL, Patrick. On tightening the relaxations of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems. *Operations Research*, v. 50, n. 4, p. 656-669, 2002.

[15] VERFAILLIE, Gérard; LEMAÎTRE, Michel; SCHIEX, Thomas. Russian doll search for solving constraint optimization problems. *AAAI/IAAI*, v. 1, p. 181-187, 1996.

[16] WOLSEY, Laurence. Integer programming. New York: Wiley, 1998.

[17] `http://www.sagemath.org/`

[18] `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`