UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Fabio Carlos Sousa Dias

Problema de Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos

FORTALEZA, CEARÁ

2014

Fabio Carlos Sousa Dias

Problema de Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos

Tese apresentada à Coordenação do Programa de Mestrado e Doutorado em Ciência da Computação, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração:Otimização

Orientador: Manoel Campêlo Bezerra Neto

FORTALEZA, CEARÁ

2014

Dados Internacionais de Catalogação na Publicação (CIP)
Universidade Federal do Ceará
Biblioteca da Matemática

A000z Dias, Fabio Carlos Sousa.

Problema de Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos / Fabio Carlos Sousa Dias. – Fortaleza, 2014.

132p.;il.

Tese(Doutorado em Ciência da Computação) - Universidade Federal do Ceará, Centro de Ciência.

Orientador: Prof. Dr. Manoel Campêlo Bezerra Neto

Área de Concentração: Otimização

1. Árvore Geradora Mínima com Restrição de Grau 2. Heurísticas 3. Programação Inteira 4. Otimização I. Universidade Federal do Ceará, Centro de Ciência.

CDD:000.0

Fabio Carlos Sousa Dias

Problema de Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos

Tese apresentada à Coordenação do Programa de Mestrado e Doutorado em Ciência da Computação, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação. Área de Concentração: Otimização

Aprovada em: __/__/____

BANCA EXAMINADORA

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

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

Prof. Dr. Críston Pereira de Souza
Universidade Federal do Ceará - UFC

Prof. Dr. Luiz Satoru Ochi
Universidade Federal Fluminense - UFF

Prof. Dr. Maurício Cardoso de Souza
Universidade Federal de Minas Gerais - UFMG

AGRADECIMENTOS

O meu primeiro agradecimento vai para aquele que esteve sempre comigo nessa jornada acadêmica, desde a luta para passar no vestibular, terminar a Graduação, o Mestrado e por fim o Doutorado, que nos momentos difíceis, de desânimo, não me deixou desistir, me fortaleceu e me guiou até esse momento. Ao meu Senhor Jesus, todo agradecimento, honra e louvor por tudo que ele fez e está fazendo em minha vida.

Irei repetir uma parte do agradecimento da minha dissertação de mestrado por considerar que representa bem meus sentimentos: ”Agradeço a todos os meus familiares por proporcionarem a minha educação e formação, em especial, aos meus avôs que me criaram como um verdadeiro filho, com muito amor e carinho, que mesmo na dificuldade se empenharam em proporcionar condições financeiras para eu entrar na faculdade. Em momória, ao meu avô que me deu não apenas condições financeiras para chegar até aqui, mais também, moral e confiança. Minha avó pelo incetivo e pela confiança, minha mãe pelo apoio e incentivo durante esse caminho.”Ao meu irmão que contribuiu quando não passei no primeiro vestibular e meus avôs não tinham mais condições de arcar com o cursinho.

Um agradecimento especial ao meu orientador Prof. Manoel Campêlo pela sua orientação acadêmica, desde o tempo de iniciação acadêmica (com aquela bolsa de IC que salvou minha vida), que continuou no mestrado e estamos agora no doutorado. Não digo finalizando, pois espero continuar a cooperação, pesquisando e trabalhando juntos. Espero continuar aprendendo e crescendo com seu conhecimento.

Agradeço a todos os professores da Universidade Federal do Ceará (UFC) pela formação acadêmica, e em especial a todos os professores do ParGo (Paralelismo, Grafos e Otimização) como o Prof. Rafael Andrade que contribuiu com o meu trabalho no doutorado. A todos os alunos do ParGo que conheci no final do doutorado.

Enfim, agradeço a todos que direta ou indiretamente colocaram um tijolo em minha formação acadêmica e com isso contribuíram para que eu chegasse até aqui.

”E, se alguém me servir, o Pai o honrará...”
Jesus Cristo

RESUMO

O Problema de Árvore Geradora Mínima com Restrição de Grau Mínimo (*Min-Degree Constrained Minimum Spannig Tree* - MD-MST) consiste em encontrar uma árvore geradora mínima de um grafo onde cada vértice ou é folha da árvore ou satisfaz uma restrição de grau mínimo. Os vértices folhas são chamados terminais e os demais são os centrais. Definimos e estudamos uma variação desse problema, que denotamos MDF-MST, onde os terminais e centrais são definidos a priori. Mostramos que o problema é NP-Difícil e está na Classe FPT, parametrizado pelo número de centrais. Identificamos também casos onde o problema torna-se polinomial. Propomos várias formulações de programação inteira para o problema e comparamos teórica e computacionalmente a qualidade do limite inferior gerado por suas relaxações lineares. Propomos e testamos uma relaxação lagrangeana para o problema, que usamos também para definir heurísticas lagrangenas. Definimos heurísticas gulosas, uma busca VND e uma heurística VNS. Apresentamos uma decomposição de Benders. Propomos uma nova heurística geral que combina ingredientes da decomposição de Benders com método de subgradientes, a qual denominamos Heurística de Subgradientes. Aplicamos tal heurística ao MDF-MST. Todos esses algoritmos foram implementados, testados, comparados entre si e com o solver CPLEX. A eficiência computacional dos algoritmos propostos, especialmente a relaxação lagrangeana, é competitiva com a do CPLEX, e superior em vários casos. Alguns desses algoritmos foram adaptados para o problema MD-MST e seu correlato DC-MST (este último onde a restrição sobre os centrais é de grau máximo). Quando comparamos os resultados computacionais com a literatura, podemos concluir que os algoritmos são competitivos.

Palavras-chave: Árvore Geradora Mínima com Restrição de Grau, Heurísticas, Programação Inteira, Otimização

ABSTRACT

The Min-Degree Constrained Minimum Spannig Tree - MD-MST is to find a minimum spanning tree of a graph where each vertex is a leaf of the tree or satisfies a constraint of minimum degree. The leaf vertices are called terminals and the others are the central vertices. We define and study a variation of this problem, which we denote MDF-MST, where the terminal and central vertices are fixed. We show that the problem is NP-Hard and is in FPT, parameterized by the number of central vertices. We also identify cases where the problem becomes polynomial. We propose several integer programming formulations for the problem and compare the quality of lower bound generated by their linear relaxations. We propose and teste a Lagrangian Relaxation for the problem, which we also use to define Lagrangian heuristics. We define greedy heuristics, a VND Local search and a VNS heuristic. We present a Benders's Decomposition. We propose a new general heuristic that combines ingredients from the Benders's decomposition with subgradient method, which we call subgradient heuristic. We apply this heuristic to the MDF-MST. All these algorithms have been implemented, tested and compared among them and with the CPLEX solver. The computational efficiency of the proposed algorithms, especially the Lagrangian heuristics, is comparable with that of CPLEX, and even better in several cases. Some of these algorithms were adapted for the MD-MST and DC-MST (in the latter, the degree constraint is of maximum degree). When comparing the computational results with the literature, we conclude that the algorithms are competitive.

Keywords: Degree-Constrained Minimum Spanning Tree, Heuristics, Integer Programming, Optimization

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Árvore é um tipo particular de grafo que possui grande importância em Otimização Combinatória. Tal importância justifica-se por ser uma estrutura usada para modelar inúmeros problemas ou subproblemas que envolvem conectividade.

Por definição, uma árvore é um grafo conexo e acíclico. Uma propriedade fundamental, muitas vezes usada para definir esse tipo de grafo, encontra-se no fato de que a diferença entre o número de vértices e de arestas em uma árvore é sempre 1. Isto nos leva as seguintes caracterizações:

Proposição 1.1 *Seja G um grafo com n vértices e m arestas. As seguintes propriedades são equivalentes e asseguram que G é uma árvore:*

- *G é conexo e acíclico;*
- *G é conexo e $m = n - 1$;*
- *G é acíclico e $m = n - 1$.*

Como veremos mais adiante, existem diversas formulações matemáticas que modelam árvore, cada uma usando uma das caracterizações apresentadas acima.

Dentre os vários problemas envolvendo árvore, destacamos aquele de encontrar uma árvore geradora de um grafo.

Definição 1.2 *Uma árvore geradora de um grafo G é um subgrafo gerador de G que é uma árvore, sendo um subgrafo gerador um subgrafo que contém todos os vértices do grafo e um subconjunto de suas arestas.*

O conceito de árvore geradora reveste-se de grande importância devido à sua aplicabilidade em diversos problemas práticos, principalmente em problemas relacionados a redes (de comunicação, de computadores, de transportes etc). Não é difícil percebermos esse fato quando observamos uma árvore geradora como um subgrafo que conecta todos os vértices de um grafo com a menor quantidade de arestas.

Quando a cada aresta do grafo está associado um custo ou peso, podemos definir o problema da *Árvore Geradora Mínima* (*Minimum Spanning Tree*, MST), que consiste em encontrar uma árvore geradora cuja soma dos pesos de suas arestas seja mínimo. Um dos algoritmos mais populares para encontrar uma MST é o algoritmo de Kruskal, primeiramente proposto em [34], que encontra uma solução ótima em tempo polinomial no tamanho do grafo.

Na literatura, têm-se estudado vários problemas relacionados ao MST, onde normalmente se impõe uma restrição adicional às de gerar uma árvore. Podemos citar o *Bounded Diameter Minimum Spanning Tree* [1], *Degree Constrained Minimum Spanning Tree* [44],

Capacitated Minimum Spanning Tree [23], *Generalized Minimum Spanning Tree* [43], *Delay-Constrained Minimum Spanning Tree* [48], *Hop-Constrained Minimum Spanning Tree* [28], *Maximum Leaf Spanning Tree* [26], como alguns exemplos de problemas conhecidos relacionado ao MST.

Entre esses problemas relacionados ao MST, vamos enfatizar os problemas cuja restrição adicional é definida sobre o grau de cada vértice (número de ligações do vértice) na árvore geradora. Para cada tipo de restrição ao grau dos vértices, define-se um problema específico. Vamos destacar alguns problemas desse grupo.

O problema da Árvore Geradora com Restrição de Grau (Máximo) (*Degree-Constrained Spanning Tree Problem*, DC-ST) e o problema da Árvore Geradora com Restrição de Grau Mínimo (*Min-Degree Constrained Spanning Tree*, MD-ST) consistem em encontrar uma árvore geradora de um grafo que respeite as restrições de grau máximo e mínimo, respectivamente, que são impostas aos seus vértices. Vale lembrar que toda árvore possui pelo menos dois vértices de grau 1, ou seja, folhas. Assim, o MD-ST busca uma árvore geradora onde cada vértice ou é folha ou satisfaz uma restrição de grau mínimo.

Em ambos os problemas, DC-ST e MD-ST, não existe custo associado às arestas. Assim, desejamos encontrar qualquer árvore geradora que respeite a restrição de grau máximo/mínimo. Em outras palavras, trata-se de problemas de decisão.

Quando consideramos um grafo ponderado e desejamos encontrar uma árvore geradora mínima que respeite as restrições de grau dos vértices, estamos diante do problema da Árvore Geradora Mínima com Restrição de Grau (Máximo) (*Degree-Constrained Minimum Spanning Tree Problem*, DC-MST) e do problema da Árvore Geradora Mínima com Restrição de Grau Mínimo (*Min-Degree Constrained Minimum Spanning Tree*, MD-MST). Note a inclusão da leta "M"(de mínimo) nas siglas referentes aos problemas ponderados.

O problema DC-MST tem sido bastante estudado (ver, por exemplo, as referências [44, 12, 15, 7, 16, 17, 5, 6]). Já o problema MD-MST vem ganhando destaque mais recentemente na literatura com diversos trabalhos relacionados [3, 4, 40, 41, 38].

Neste trabalho iremos abordar os problemas MD-MST e DC-MST e, principalmente, uma variação do MD-MST que definimos. Nessa variação, os vértices folha e os centrais (aqueles que devem obedecer a restrição de grau mínimo) são fixados a priori, diferentemente do que ocorre no MD-MST, onde todo vértice pode ser escolhido como central ou folha. Essa variação será denotada por MDF-MST (o "F"refere-se a fixo). Iremos provar que o problema é NP-Difícil, apresentar propriedades, algumas formulações e diversos algoritmos.

1.1 Organização do Texto

Além deste capítulo introdutório, esta tese está estruturada em mais 7 capítulos, cujo conteúdo descrevemos a seguir.

Formulações para os problemas derivados de MST com adição de restrição (por exemplo, de grau) podem ser obtidas por formulações que definem o MST mais as restrições

particulares de cada problema. Sendo assim, no Capítulo 2, revisamos algumas formulações matemáticas de árvore geradora, em especial as que são usadas no desenvolvimento deste trabalho. Iremos apresentar as tradicionais formulações baseadas nas restrições de eliminação de sub-rotas (*Subtour Elimination Constraints*, SECs), introduzidas por Dantzig et al. [18], e nas restrições de corte direcionadas (*Directed Cutset Constraints*, DCUTs). Essas duas formulações se caracterizam como sendo formulações com um grupo de restrições em quantidade exponencial. Também iremos considerar formulações compactas, ou seja, com quantidade de restrições e variáveis polinomial. Dentre essas formulações compactas, destacamos uma que tem ganho destaque na literatura recente, proposta por Martin [37], que visualiza a árvore geradora como a intersecção de n árvores geradoras enraizadas em cada vértice. Também iremos descrever formulações baseadas em restrições de fluxo simples e multifluxo e nas restrições de Miller-Tucker-Zemlin [42]. Terminamos o capítulo com uma breve comparação entre as formulações, com respeito às suas dimensões e aos limites de programação linear que podem fornecer, o que nos foi útil para definir quais delas vamos usar no seguimento do trabalho.

No Capítulo, 3 apresentamos formalmente o problema-alvo deste trabalho. Esse problema é uma variação do MD-MST, que iremos chamar de Problema da Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos (MDF-MST), que ocorre quando os vértices centrais e terminais são definidos a priori. Mostramos que o problema é NP-Difícil e está na Classe FPT ([20],[21]), parametrizado pelo número de centrais. Identificamos também casos onde o problema torna-se polinomial, além de várias propriedades relacionadas à sua viabilidade.

No Capítulo 4, iremos apresentar diversas formulações para o MDF-MST, cada uma baseada em uma das formulações do MST apresentadas no Capítulo 2. Iremos realizar uma avaliação teórica e computacional do limite de programação linear destas formulações para o MDF-MST, para analisarmos a qualidade da relaxação linear de cada formulação. Iremos propor Algoritmos de Planos de Corte para as formulações com quantidade de restrições exponencial. Para realizarmos os experimentos computacionais, utilizaremos instâncias da literatura propostas para o MD-MST. Iremos também propor outras instâncias de teste, potencialmente mais difíceis para o MDF-MST, que serão utilizadas no restante do trabalho.

No Capítulo 5, apresentamos diversos algoritmos para o MDF-MST. São algoritmos exatos e heurísticos. Inicialmente, utilizamos o solver CPLEX com as instâncias construídas, procurando avaliar a eficiência deste solver e das formulações propostas para o MDF-MST. Depois propomos e avaliamos computacionalmente diversas heurísticas. Elaboramos heurísticas gulosas, baseadas em trocas de arestas e em busca em vizinhança como o VND, uma heurística VNS e heurísticas lagrangeanas. Essa última baseia-se na Relaxação Lagrangeana que utilizamos tanto para gerar limites superiores (através das heurísticas) quanto inferiores (através do método de subgradientes) para o MDF-MST. Por fim, implementamos uma Decomposição de Benders. A eficiência computacional de todos esses algoritmos será comparada através de suas aplicações às instâncias de teste.

Neste Capítulo, propormos uma nova heurística geral que combina ingredientes da Decomposição de Benders com o Método de Subgradientes, a qual denominaremos Heurística de Subgradientes. Adiantamos que um passo fundamental dessa heurística pode ser implemen-

tado de formas diferentes para um mesmo problema e, por questões de tempo, nossos experimentos não foram exaustivos.

Todo esse ferramental desenvolvido para o MDF-MST, pode ser adaptado, com pequeno esforço, para aplicação ao DC-MST. Este será o conteúdo do Capítulo 6, cujo propósito será avaliar o desempenho desses algoritmos frente aos resultados da literatura. Em outras palavras, queremos avaliar como a aplicacação quase direta desses algoritmos propostos para o MDF-MST, sem a incoporação de outros elementos especificamente direcionados ao DC-MST, comporta-se para este último problema. Para tornar as formulações dos dois problemas mais semelhantes, particionamos também os vértices de uma instância de DC-MST em centrais e terminais, sendo os últimos aqueles que têm grau máximo igual a 1 e, portanto, serão folhas. Particularizaremos nossa análise na aplicação da Relaxação Lagrangeana, que foi abordagem de melhor desempenho para o MDF-MST. As instâncias de teste neste caso são aquelas usadas na literatura para MD-MST, de modo a ser possível uma comparação.

Por fim, no Capítulo 7, resumimos nossas contribuições e apresentamos algumas conclusões e direções para trabalhos futuros.

1.2 Notações

Ao longo do texto, iremos introduzindo as notações necessárias no momento. Aqui, apresentaremos apenas notações gerais, comuns a textos que lidam com grafos.

Iremos considerar um grafo $G = (V,E)$, onde V é o conjunto de vértices, com $|V| = n$, e E o conjunto de arestas, com $|E| = m$. Sejam V_1 e V_2 dois subconjuntos (não necessariamente distintos) de V. Iremos denotar por $\delta(V_1 : V_2) = \{e = \{i, j\} \in E : i \in V_1 \text{ e } j \in V_2\}$ o conjunto de arestas com uma extremidade no conjunto V_1 e a outra extremidade no conjunto V_2. Em particular, $\delta(V_1) = \delta(V_1 : \overline{V_1})$ é o conjunto de arestas com uma extreminade em V_1 e outra em seu complemento $\overline{V_1} = V \backslash V_1$. Em outras palavras, $\delta(V_1)$ é o conjunto de arestas no corte $(V_1, \overline{V_1})$. Por simplicidade de notação, quando $V_1 = \{v\}$ iremos denotar o corte $\delta(V_1)$ apenas como $\delta(v)$. Já $E(V_1) = \delta(V_1 : V_1)$ irá expressar o conjunto de arestas com ambas as extremidades em V_1, ou seja, o conjunto das arestas de G induzidas por V_1. O subgrafo induzido por V_1 será denotado $G(V_1) = (V_1, E(V_1))$. No caso de o grafo G não estar claro pelo contexto, usaremos o subíndice $_G$ para indetificá-lo nas notações acima.

Quando considerarmos um grafo direcionado, iremos usar $D = (V,A)$, sendo A o conjunto das arestas direcionadas ou arcos. Iremos extender as definições acima, substituindo o termo aresta(s) por arco(s). Além disso, particionamos cada um dos conjuntos assim definidos em dois, separando as arestas com origem ou destino no conjunto argumento. Assim, $\delta^+(V_1 : V_2)$ (resp. $\delta^-(V_1 : V_2)$) é o conjunto de arcos de $\delta(V_1 : V_2)$ com origem (resp. destino) em V_1. Similarmente, definimos $\delta^+(V_1) = \delta^+(V_1 : \overline{V_1})$ e $\delta^-(V_1) = \delta^-(V_1 : \overline{V_1})$. Também iremos considerar os cortes de entrada e saída de um único vértice v como $\delta^+(v)$ e $\delta^-(v)$, respectivamente.

Tanto para um grafo não direcionado como para um direcionado, $d(v) = |\delta(v)|$ é o grau de um vértice v. Analogamente, definiremos $d^+(v)$ e $d^-(v)$ como grau de saída e de

entrada, respectivamente. Novamente, um subíndice identificando o grafo de referência pode ser usado nessa notação. O grau máximo de um vértice em G será denotado $\Delta(G) = \max\{d(v) : v \in V\}$.

Durante todo o restante do texto, iremos tratar de problema de minimização. Com isso, sempre que falarmos sobre um método ou ferramenta, estaremos estritamente escrevendo relacionando o método ou ferramenta a um problema de minimização. Por exemplo, quando tratarmos de limite inferior e superior, estaremos nos referindo a uma relaxação/dual e a uma solução viável do problema.

2 FORMULAÇÕES MATEMÁTICAS PARA ÁRVORE GERADORA MÍNIMA

Ao longo desse capítulo, consideramos um grafo conexo e não-direcionado $G = (V, E)$, com n vértices, m arestas, e com custo $c_e \in \mathbb{R}$ associado a cada aresta $e \in E$. O *Problema de Árvore Geradora Mínima* (MST) deseja encontrar um subgrafo gerador de G de menor custo que seja uma árvore. Como já mencionamos anteriormente, esse problema pode ser resolvido de forma ótima e em tempo polinomial, utilizando, por exemplo, o algoritmo de Kruskal [34].

Neste capítulo, iremos apresentar as principais formulações matemáticas existentes na literatura para o MST. Não iremos ser exaustivos nas explicações dos modelos. Boa parte das formulações aqui apresentadas e comparações entre suas relaxações lineares podem ser encontradas em Magnanti e Wolsey [36].

Como vimos, um subgrafo é uma árvore geradora se satisfaz duas entre as três seguintes condições: (i) o número de arestas é um a menos que o número de vértices do grafo; (ii) é conexo, (iii) não possui ciclos. As formulações que apresentamos garantem de forma explícita ou ímplicita tais propriedades.

2.1 Eliminação de Ciclos ou Cortes

2.1.1 Árvore não orientada

Uma formulação para árvore geradora pode ser obtida utilizando as restrições de eliminação de ciclos ou, como mais comumente conhecidas, devido ao seu uso no Problema de Caixeiro Viajante, restrições de eliminação de sub-rotas (*Subtour Elimination Constraints*, SECs), introduzidas por Dantzig et al. [18]. Como sugere o nome, essas restrições proíbem a ocorrência de ciclos no subgrafo gerado, condição necessária para a obtenção de uma árvore.

Considere abaixo a formulação de programação inteira para o MST, onde temos a variável binária x_e sendo igual a 1 se a aresta $e \in E$ for escolhida para pertencer a árvore geradora e 0 caso contrário:

$$(\text{MST}_{\text{subtour}}) \quad \min \sum_{e \in E} c_e x_e \tag{2.1}$$

$$\text{sujeito a:} \sum_{e \in E} x_e = n - 1 \tag{2.2}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \forall S \subsetneq V \text{ com } |S| \geq 2 \tag{2.3}$$

$$x_e \in \{0,1\}, \forall e \in E \tag{2.4}$$

A função objetivo (2.1) minimiza o custo da árvore geradora. A restrição (2.2) garante que teremos $n-1$ arestas escolhidas. O conjunto de restrições (2.3), as SECs, garante que o grafo gerado não terá ciclos. Assim, as restrições (2.2) e (2.3) definem uma árvore.

Note que o conjunto de restrições (2.3) tem tamanho exponencial no tamanho do grafo. Em particular, quando $|S| = 2$, elas correspondem a $x_e \leq 1$, $\forall e \in E$.

Vamos denotar por P_{subtour} o politopo associado à relaxação linear da formulação $\text{MST}_{\text{subtour}}$, ou seja, o conjunto dos pontos $x \geq 0$ que satisfazem (2.2)-(2.3). As restrições $x_e \geq 0, \forall e \in E$, definem facetas de P_{subtour}. Também definem facetas as restrições (2.3) quando $|S| = 2$ e os subgrafos $G(S)$ and $G(V \backslash S)$ forem conexos ou quando $|S| \geq 3$ e o grafo $G(S)$ for 2-conexo. Em particular, todas essas condições são satisfeitas se G for um grafo completo.

Podemos simplificar a formulação acima, considerando as restrições de eliminação de ciclos (2.3) apenas para os subconjuntos de vértices que definem ciclos (não necessariamente induzidos) no grafo, ou seja,

$$\sum_{e \in E(W)} x_e \leq |W| - 1, \forall W \subsetneq V : W \text{ define um ciclo em } G$$

A formulação (2.1)–(2.4) usa a definição de árvore como um grafo com $n-1$ arestas e que não contém ciclo. Quando consideramos uma árvore como um grafo conexo com $n-1$ arestas, então podemos definir uma outra formulação de programação inteira:

$$\begin{aligned}
(\text{MST}_{\text{cutset}}) \quad & \min \sum_{e \in E} c_e x_e \\
& \text{sujeito a:} \sum_{e \in E} x_e = n - 1 && (2.5) \\
& \sum_{e \in \delta(S)} x_e \geq 1, \forall S \subsetneq V \text{ com } S \neq \varnothing && (2.6) \\
& x_e \in \{0, 1\}, \forall e \in E && (2.7)
\end{aligned}$$

A única diferença entre as duas formulações se encontra nos conjuntos de restrições (2.3) e (2.6). Em (2.6), garante-se que o grafo gerado será conexo. Esse conjunto de restrições, chamadas restrições de cortes não-direcionados, também possui um número exponencial de desigualdades. Vamos denotar por P_{cutset} o conjunto viável da relaxação linear de $\text{MST}_{\text{cutset}}$, isto é, o politopo definido por $0 \leq x_e \leq 1$, $\forall e \in E$, e as resrições (2.5)–(2.6).

Quando consideramos em (2.6) um multicorte, ao invés de um corte simples, geramos uma outra formulação para o MST. Particione o conjunto V em $k+1$ subconjuntos $W_0, W_1, \ldots, W_k$ de vértices, $k \geq 1$, e defina $\delta(W_0, W_1, \ldots, W_k)$ como as arestas que cruzam esses conjuntos, ou seja, arestas com uma extremidade em um conjunto e a outra extremidade em um outro conjunto. Como toda árvore geradora é conexa, então, para qualquer partição $W_0, W_1, \ldots, W_k$ de V, teremos pelo menos k arestas ligando esses conjuntos de vértices em qualquer árvore geradora. Com isso, podemos definir a seguinte formulação de programação inteira:

$$(\text{MST}_{\text{multicut}}) \quad \min \sum_{e \in E} c_e x_e$$

$$\text{sujeito a:} \sum_{e \in E} x_e = n - 1 \tag{2.8}$$

$$\sum_{e \in \delta(W_0, W_1, \ldots, W_k)} x_e \geq k, \forall k \geq 1, \forall \text{ partição } W_0, W_1, \ldots, W_k \text{ de } V \tag{2.9}$$

$$x_e \in \{0,1\}, \forall e \in E \tag{2.10}$$

O conjunto viável da relaxação dessa formulação será denotado por P_{multicut}. Magnanti e Wolsey [36] comparam as relaxações dos modelos acima definidos.

Proposição 2.1 *$P_{subtour} = P_{multicut} \subseteq P_{cutset}$. Mais ainda, $P_{subtour}$ é um poliedro inteiro.*

2.1.2 Árvore enraizada

O problema de árvore geradora é definido sobre um grafo não orientado. Entretanto, a orientação do grafo traz alguns benefícios em termos de modelagem, podendo eliminar simetrias e fortalecer a relaxação linear. A árvore obtida será orientada a partir de um vértice escolhido como raiz.

Nesse sentido, vamos transformar o grafo não-direcionado $G = (V, E)$ em um grafo direcionado $D = (V, A)$. Substituímos cada aresta $e = (i, j) \in E$ pelos arcos (i, j) e (j, i) em A, com custo igual c_e em ambas as direções. Definida uma raiz $r \in V(G)$, os arcos escolhidos para formar a árvore são tais que o caminho entre r e cada outro vértice $v \in V \setminus \{r\}$ é orientado, com origem em r e destino em v. Para isso, para cada $(i, j) \in A$, seja y_{ij} uma variável binária que será igual a 1 se o arco (i, j) estiver na solução e 0 caso contrário.

Essa estratégia permite fortalecer a formulação $\text{MST}_{\text{cutset}}$, como segue:

$$(\text{MST}^r_{\text{cutsetD}}) \quad \min \sum_{(i,j) \in A} c_{ij} y_{ij}$$

$$\text{sujeito a:} \sum_{(i,j) \in A} y_{ij} = n - 1 \tag{2.11}$$

$$\sum_{(i,j) \in \delta^+(S)} y_{ij} \geq 1, \forall S \subsetneq V \text{ e } r \in S \tag{2.12}$$

$$y_{ij} \geq 0, \forall (i,j) \in A \tag{2.13}$$

$$x_e = y_{ij} + y_{ji}, \forall e = (i,j) \in E \tag{2.14}$$

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

Note que, para cada raiz r escolhida, temos uma formulação distinta. As restrições (2.12) são chamadas de restrições de corte direcionadas (*Directed Cutset Constraints*, DCUTs).

Para qualquer $j \in V \setminus \{r\}$, a restrição (2.12) correspondente a $S = V \setminus \{j\}$ é $\sum_{(i,j)\in\delta^-(j)} y_{ij} \geq 1$. Somando-se essas restrições e considerando (2.11), chegamos à conclusão que

$$\sum_{(i,j)\in\delta^-(j)} y_{ij} = 1, \forall j \neq r, \tag{2.16}$$

e

$$y_{ir} = 0, \forall (i,r) \in \delta^-(r), \tag{2.17}$$

são satisfeitas pelas soluções de (2.11)–(2.13). Muitas vezes estas igualdades são incluídas diretamente na formulação, especialmente quando se pensa em um processo de geração de restrições para a resolução do modelo.

A substituição das restrições de corte (2.6) pelas restrições de corte direcionadas (2.12) leva a um fortalecimento da relaxação linear, como mostrado por Magnanti e Wolsey [36]. Seja P^r_{cutsetD} a projeção sobre x do conjunto viável da relaxação linear da formulação $\text{MST}^r_{\text{cutsetD}}$, ou seja, o conjunto dos pontos x, $0 \leq x_e \leq 1$, tais que (x,y) satisfaz (2.11)–(2.14) para algum y. Então:

Proposição 2.2 *Para todo $r \in V$, $P^r_{cutsetD} = P_{subtour}$.*

Por esse resultado, podemos concluir que a integralidade das variáveis x podem ser descartadas. Por conseguinte, as restrições (2.14) podem ser substituídas por $y_{ij} + y_{ji} \leq 1$ e as variáveis x, eliminadas.

2.2 Conectividade via Coincidência de Arestas

As formulações $\text{MST}_{\text{subtour}}$ e $\text{MST}^r_{\text{cutsetD}}$ apresentam duas características fortemente desejáveis: usam apenas m variáveis (a dimensão do vetor de incidência de uma árvore geradora) e gozam da propriedade de integralidade (as restrições de integralidade das variáveis podem ser descartadas). Entretanto, ambos apresentam um número exponencial de restrições, que em geral precisam ser separadas. Embora a separação em ambos os casos seja polinomial, este precesso pode tornar-se computacionalmente dispendioso. Tal incoveniente motiva a procura por outras formulações que também possuam a propriedade da integralidade, mas que utilizem um número menor de restrições.

Esta característica apresenta-se na formulação proposta por Martin [37]. Ela pode ser vista como uma composição de n formulações $\text{MST}^r_{\text{cutsetD}}$, uma para cada possível raiz $r \in V$. Em outras palavras, cada árvore geradora de G dá origem a n árvores geradoras direcionadas, cada uma enraizada em um vértice diferente do grafo. Assim, geramos n árvores, cada uma enraizada em um vértice, mas todas elas escolhendo (coincidindo em) as mesmas arestas de G. Como na formulação $\text{MST}^r_{\text{cutsetD}}$, vamos considerar o grafo orientado $D = (V,A)$.

Para cada arco $(i,j) \in A$ e cada $k \in V \setminus \{j\}$, vamos definir a variável λ^k_{ij} que será igual 1 se (i,j) for escolhido na árvore enraizada em k, e 0 caso contrário. Se $\lambda^k_{ij} = 1$, diremos

que i é pai de j na árvore enraizada em k. Teremos também a variável x_e, indicando se a aresta $e \in E$ está na solução. Essa variável também terá a importante função de manter a igualdade entre as n árvores, ou melhor, de assegurar que elas coincidam em todas as $n-1$ arestas.

A formulação proposta em [37] pode ser escrita como:

$$
\begin{aligned}
(\mathrm{MST}_{\mathrm{inter}}) \quad \min & \sum_{(i,j)\in E} c_e x_e \\
\text{sujeito a:} & \sum_{e\in E} x_e = |V| - 1 && (2.18)\\
& x_e = \lambda^k_{ij} + \lambda^k_{ji}, : \forall k \in V, \forall e = (i,j) \in E && (2.19)\\
& \sum_{(i,j)\in\delta^-(j)} \lambda^k_{ij} = 1, \forall j \in V, \forall k \in V, k \neq j && (2.20)\\
& \lambda^k_{ik} = 0, : \forall (i,k) \in A && (2.21)\\
& x_e \in \{0,1\}, \forall e \in E, :: \lambda^k_{ij} \in \{0,1\}, \forall (i,j) \in A, \forall k && (2.22)
\end{aligned}
$$

A restrição (2.19) indica que, se uma aresta (i,j) for escolhida para estar na solução, então, para todo $k \in V$, a árvore enraizada em $k \in V$ contém o arco (i,j) ou (j,i). As restrições (2.20) e (2.21), respectivamente, garantem que cada vértice j terá exatamente um pai na árvore enraizada em $k \neq j$ e não terá pai na árvore com raiz em $k = j$. Note que, para cada k, essas restrições são exatamente (2.16) e (2.17).

Podemos mostrar que as restrições (2.18)-(2.22) são suficientes para a definição de uma árvore geradora, o que implica a corretude da formulação. Considere o subgrafo de G induzido pelas arestas e com $x_e = 1$. Seja H uma componente conexa desse subgrafo. Tome $k \in V(H)$ e considere o subgrafo T^k de D induzido pelos arcos (i,j) tal que $i,j \in V(H)$ e $\lambda^k_{ij} = 1$, ou seja, o subgrafo de $D(V(H))$ associado à raiz k e definido pelos arcos escolhidos conforme as variáveis λ. Por (2.19) e (2.22), T^k é uma orientação do subgrafo conexo H. E pelas restrições (2.21)–(2.20) relativas a k, todo vértice do subgrafo conexo T^k tem exatamente um pai, exceto por k, que não tem pai. Logo, T^k é uma árvore enraizada em k, o que implica que H é uma árvore. Assim, temos que o subgrafo definido pelas variáveis x tem $n-1$ arestas (por (2.18)) e que suas componentes conexas são árvores. Logo, esse subgrafo é uma árvore.

Gouveia e Telhada [29] apresentaram uma outra formulação para definir árvores geradoras, que eles chamaram de Reformulação por Interseção. Essa reformulação é apenas a aplicação da ideia proposta por Martin [37] para transformar formulações com muitas restrições em formulações compactas, com base nos algoritmos de separação associados. Os próprios autores em [30] utilizaram o resultado em [37] para provar a corretude de sua formulação.

Seja P_{inter} a projeção do conjunto viável da relaxação linear de MST_{inter} sobre x. Martin [37] mostrou que esse poliedro é inteiro (ver também [14]).

Proposição 2.3 $P_{inter} = P_{subtour}$.

2.3 Conectividade via Fluxo

Uma alternativa para modelar a conectividade requerida pela árvore geradora, usando um número polinomial de restrições, encontra-se na estratégia de enviar um fluxo através do grafo direcionado $D = (V,A)$. Com isso, dois novos tipos de formulações são obtidos para o MST: Formulação de Fluxo Simples (*single commodity*) de Multifluxo (*multicommodities*). Nestas duas formulações, continuaremos a trabalhar com a variável binária x_e, que irá indicar se a aresta $e = (i, j) \in E$ será escolhida ou não para formar a árvore e, portanto, se poderá ser usada ou não para transportar fluxo entre i e j. Teremos também, para arco $(i, j) \in A$, uma variável para descrever o valor do fluxo a ser transportado de *i para* j. Como nas formulações direcionadas anteriores, vamos escolher a priori um vértice fonte (raiz), digamos r, a partir do qual o fluxo se origina.

Para a formulação de fluxo simples, definimos a variável de fluxo f_{ij}, que irá armazenar o valor fluxo no arco (i, j), na direção do vértice i para o vértice j; assim, para cada aresta $e = (i, j) \in E$, teremos as variáveis f_{ij} e f_{ji}. Com isso, obtemos:

$$(\text{MST}^r_{\text{flow}}) \quad \min \sum_{e \in E} c_e x_e$$

$$\text{sujeito a:} \sum_{(r,j) \in \delta^+(r)} f_{rj} - \sum_{(i,r) \in \delta^-(r)} f_{ir} = n - 1 \tag{2.23}$$

$$\sum_{(i,j) \in \delta^-(j)} f_{ij} - \sum_{(j,\ell) \in \delta^+(j)} f_{j\ell} = 1, \forall j \in V \backslash \{r\} \tag{2.24}$$

$$f_{ij} \leq (n-1) x_e, \forall e = (i, j) \in E \tag{2.25}$$

$$f_{ji} \leq (n-1) x_e, \forall e = (i, j) \in E \tag{2.26}$$

$$\sum_{e \in E} x_e = n - 1 \tag{2.27}$$

$$f \geq 0, x_e \in \{0, 1\}, \forall e \in E \tag{2.28}$$

A restrição (2.23) obriga que $n-1$ unidades de fluxo saiam do vértice fonte r. As restrições (2.24) garantem que cada vértice será atendido com uma única unidade de fluxo. Essas restrições asseguram a conectividade da solução ao estabelecer um caminho entre a raiz e cada outro vértice da rede através do fluxo enviado. Em outras palavras, elas garantem o balanceamento do fluxo, por isso são chamadas de restrições de balanceamento. Na verdade, (2.23) é exatamente a soma de todas as restrições em (2.24), sendo, portanto, redundante na formulação.

As restrições (2.25) e (2.26) permitem que uma aresta seja utilizada para transportar fluxo, em alguma direção, apenas se ela tiver sido escolhida. Já a restrição (2.27) garante que $n-1$ arestas serão escolhidas e, juntamente com as restrições de balanceamento, garantem que a solução será uma árvore geradora.

Magnanti e Wolsey [36] mostram que a relaxação linear da formulação acima pode fornecer um limite inferior bem fraco. Uma maneira de fortalecer esse limite é usar a ideia de multifluxo. Diferentemente do fluxo simples, onde as $n-1$ unidades de fluxo saem da fonte

sem uma destinação específica, agora cada uma das $n-1$ unidades de fluxo é endereçada para um vértice particular. Para isso, precisaremos da variável de fluxo f_{ij}^k, que irá armazenar o fluxo no arco (i,j) com destino ao vértice $k \neq r$. Assim obtemos:

$$(\text{MST}^r_{\text{multiflow}}) \quad \min \sum_{e \in E} c_e x_e$$

$$\text{sujeito a:} \sum_{(r,j)\in\delta^+(r)} f_{rj}^k - \sum_{(i,r)\in\delta^-(r)} f_{ir}^k = 1, \forall k \neq r \tag{2.29}$$

$$\sum_{(i,j)\in\delta^-(j)} f_{ij}^k - \sum_{(j\ell)\in\delta^+(j)} f_{j\ell}^k = 0, \forall j \in V\backslash\{r,k\}, \forall k \neq r \tag{2.30}$$

$$\sum_{(i,k)\in\delta^-(k)} f_{ik}^k - \sum_{(k,j)\in\delta^+(k)} f_{kj}^k = 1, \forall k \neq r \tag{2.31}$$

$$f_{ij}^k \leq y_{ij}, \forall (i,j) \in A, \forall k \neq r \tag{2.32}$$

$$\sum_{(i,j)\in A} y_{ij} = n-1 \tag{2.33}$$

$$x_e = y_{ij} + y_{ji}, \forall e = (i,j) \in E \tag{2.34}$$

$$f \geq 0,\ y_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{2.35}$$

$$x_e \in \{0,1\}, \forall e \in E \tag{2.36}$$

A variável y_{ij} é a mesma das formulações anteriores. Como y_{ij} é binária, a restrição (2.32) indica que cada arco (i,j) poderá transportar no máximo uma unidade de cada *commodity*, se o arco for alocado. Em outras palavras, $y_{ij} \in \{0,1\}$ é a capacidade do arco (i,j). Da mesma forma que no caso simples, as restrições de balanceamento (2.29), (2.30) e (2.31) garantem que a solução seja conexa.

Como antes, para cada k, a soma das restrições (2.30)–(2.31) associadas a k resulta exatamente na restrição de (2.29) referente a k. Logo, as restrições (2.29) podem ser suprimidas. Adicionalmente, por (2.32), (2.31) e $f \geq 0$, observe que

$$\sum_{(i,k)\in\delta^-(k)} y_{ik} \geq \sum_{(i,k)\in\delta^-(k)} f_{ik}^k \geq 1,\ \forall k \neq r.$$

Então, por (2.33), para todo $k \neq r$ deve acontecer $\sum_{(i,k)\in\delta^-(k)} y_{ik} = 1$ e, portanto, $\sum_{(i,k)\in\delta^-(k)} f_{ik}^k = 1$. Com isso, as equações (2.32) podem ser substituídas por

$$\sum_{(i,k)\in\delta^-(k)} f_{ik}^k = 1, \forall k \neq r. \tag{2.37}$$

Podemos perceber as restrições (2.30) e (2.31) como uma desagregação da restrição (2.24) para o caso simples, já que $f_{ij} = \sum_{k\in V\backslash\{r\}} f_{ij}^k$. E como $|V\backslash\{r\}| = n-1$, também podemos ver as restriçoes (2.32) como uma desagregação e fortalecimento de (2.26)–(2.27).

Sejam P^r_{flow} e $P^r_{\text{multiflow}}$, respectivamente, as projeções sobre x dos conjuntos viáveis das relaxações lineares de $\text{MST}^r_{\text{flow}}$ e $\text{MST}^r_{\text{multiflow}}$. Magnanti e Wolsey[36] mostram a seguinte relação entre esses conjuntos.

Proposição 2.4 *Para todo* $r \in V$, $P^r_{multiflow} = P_{subtour} \subseteq P_{cutset} \subseteq P^r_{flow}$

Na verdade, é possível obter diferentes formulações do tipo fluxo simples (com variáveis f_{ij}) e multifluxo (com variáveis f^k_{ij}), usando apenas as variáveis x_e ou y_{ij} ou ambas, o que implica a possibilidade de construção de restrições variadas para garantir a correta relação entre as variáveis de fluxo e as variáveis que definem a árvore. Em particular, apresentamos duas formulações recentemente sugeridas por Almeida, Martins e Souza [3], que se baseiam nos modelo $\text{MST}^r_{\text{flow}}$ e $\text{MST}^r_{\text{multiflow}}$. Essas formulações foram usadas em conjunto com restrições de grau mínimo, que também aparecem em nosso trabalho.

A seguir, a formulação via fluxo simples.

$$
\begin{aligned}
(\text{MST}^r_{\text{flowD}}) \quad & \min \sum_{(i,j)\in A} c_{ij} y_{ij} \\
& \text{sujeito a:} \sum_{(i,j)\in\delta^-(j)} y_{ij} = 1, \forall j \in V\backslash\{r\} && (2.38)\\
& y_{ir} = 0, \forall (i,r) \in A && (2.39)\\
& \sum_{(i,j)\in\delta^-(j)} f_{ij} - \sum_{(j\ell)\in\delta^+(j)} f_{j\ell} = 1, \forall j \in V\backslash\{r\} && (2.40)\\
& y_{ij} \leq f_{ij} \leq (n-1) y_{ij}, \forall (i,j) \in A && (2.41)\\
& f_{ij} \geq 0, y_{ij} \geq 0, \forall (i,j) \in A && (2.42)\\
& x_{ij} = y_{ij} + y_{ji}, \forall (i,j) \in E && (2.43)\\
& x_e \in \{0,1\}, \forall e \in E && (2.44)
\end{aligned}
$$

Repare que, das restrições de balanceamento (2.23) e (2.24) de $\text{MST}^r_{\text{flow}}$, apenas (2.24) foi mantida em $\text{MST}^r_{\text{flowD}}$, agora identificada como (2.40). Na verdade, já vimos que (2.23) é redundante na formulação. As restrições (2.41) e (2.43) implicam (2.25) e (2.26). Já (2.38)-(2.39) e (2.43) levam a (2.27). Portanto, se (x,y) é viável para a relaxação linear de $\text{MST}^r_{\text{flow}}$ também o é para a relaxação linear de $\text{MST}^r_{\text{flowD}}$, o que garante a corretude dessa última formulação.

Note que as variáveis x podem ser eliminadas do modelo, se impomos integralidade em y e colocamos $y_{ij} + y_{ji} \leq 1$. Elas foram mantidas aqui apenas para manter a uniformidade com as outras formulações.

Agora apresentamos a formulação baseada em multifluxo.

$$(\text{MST}^r_{\text{multiflowD}}) \quad \min \sum_{(i,j)\in A} c_{ij}y_{ij}$$

$$\text{sujeito a:} \sum_{(i,j)\in\delta^-(j)} y_{ij} = 1, \forall j \in V\backslash\{r\} \tag{2.45}$$

$$y_{ir} = 0, \forall (i,r) \in A \tag{2.46}$$

$$\sum_{(i,j)\in\delta^-(j)} f^k_{ij} - \sum_{(j,\ell)\in\delta^+(j)} f^k_{j\ell} = 0, \forall j \in V\backslash\{r,k\}, \forall k \neq r \tag{2.47}$$

$$\sum_{(i,j)\in\delta^-(j)} f^j_{ij} = 1, \forall j \in V\backslash\{r\} \tag{2.48}$$

$$f^k_{ij} \leq y_{ij}, \forall i \in V, \forall j,k \in V\backslash\{i,r\} \tag{2.49}$$

$$f^k_{ij} \geq 0, y_{ij} \geq 0, \forall (i,j) \in A, \forall k \in V\backslash\{r\} \tag{2.50}$$

$$x_{ij} = y_{ij} + y_{ji}, \forall (i,j) \in E \tag{2.51}$$

$$x_e \in \{0,1\}, \forall e \in E \tag{2.52}$$

Vamos mostrar a corretude de $\text{MST}^r_{\text{multiflowD}}$ através de sua relação com $\text{MST}^r_{\text{multiflow}}$. Pela redundância de (2.29) e por (2.37), concluímos que (2.47)-(2.48) implicam (2.29)–(2.31). Já (2.49) e (2.32) são equivalentes, enquanto (2.45)-(2.46) somadas resultam em (2.33). Estas relações levam à corretude da formulação, bem como à constatação de que a relaxação linear de $\text{MST}^r_{\text{multiflowD}}$ é pelo menos tão forte quanto a de $\text{MST}^r_{\text{multiflow}}$.

Sejam P^r_{flowD} e $P^r_{\text{multiflowD}}$ as projeções sobre x dos conjunto viáveis das relaxações lineares de $\text{MST}^r_{\text{flowD}}$ e $P^r_{\text{multiflowD}}$, respectivamente. Pelo exposto, temos

Proposição 2.5 *Para todo $r \in V$, $P^r_{multiflowD} = P^r_{multiflow} = P_{subtour} \subseteq P^r_{flowD} \subseteq P^r_{flow}$.*

Vale destacar que, diferentemente dos casos onde se usa multifluxo, a qualidade dos limites gerados pelas relaxações lineares dos modelos baseados em fluxo simples é afetada pela escolha da raiz. Isto porque pode ocorrer $P^r_{\text{flowD}} \neq P^{r'}_{\text{flowD}}$ e $P^r_{\text{flow}} \neq P^r_{\text{flow}}$ quando $r \neq r'$.

2.4 Eliminação de Ciclos via Rótulos

O intuito de obter formulações fortes para MST com um número polinomial de restrições foi obtido nas seções 2.2 e 2.3 às custas de um aumento do número de variáveis. Nesta seção, apresentamos uma forma de modelar a eliminação de ciclos, usando apenas um número linear de variáveis, em adição às variáveis x que marcam as arestas escolhidas.

A formulação, apresentada por Gouveia [27], se baseia nas restrições Miller-Tucker-Zemlin [42], comumente usadas na modelagem do Problema do Caixeiro Viajante e problemas relacionados. Basicamente, a ideia pode ser empregada em quaisquer problemas onde ciclos não são permitidos. A estratégia é determinar rótulos para os vértices e escolher um arco (i,j) apenas se o rótulo de j for maior que o rótulo de i. Isto vai assegurar que, em qualquer caminho

formado, os vértices estão dispostos em ordem crescente de rótulos, o que evita a formação de ciclos.

Dessa forma, vamos trabalhar sobre o grafo orientado $D = (V,A)$, eleger uma raiz r e, além das variáveis y_{ij} que temos usado, vamos também definir uma variável não-negativa u_i, para cada $i \in V$, que representará o rótulo de i. Para o Problema do Caixeiro Viajante, onde existe um único caminho na solução, um rótulo será atribuído de forma exclusiva para um vértice. Já no caso de MST, um mesmo rótulo pode ser usado em diferentes vértices.

A referida formulação é dada por:

$$
\begin{aligned}
(\mathrm{MST}^r_{\mathrm{MTZ}}) \quad & \min \sum_{(i,j)\in A} c_{ij} y_{ij} \\
& \text{sujeito a:} \sum_{(i,j)\in \delta^-(j)} y_{ij} = 1, \forall j \in V \backslash \{r\} && (2.53)\\
& u_i - u_j + n y_{ij} \leq n-1, \forall (i,j) \in A, j \neq r && (2.54)\\
& 1 \leq u_i \leq n-1, \forall i \in V \backslash \{r\} && (2.55)\\
& u_r = 0 && (2.56)\\
& x_{ij} = y_{ij} + y_{ji}, \forall (i,j) \in E && (2.57)\\
& x_e \in \{0,1\}, \forall e \in E,\ y_{ij} \in \{0,1\}, \forall (i,j) \in A && (2.58)
\end{aligned}
$$

A restrição (2.53) garante que, para todo vértice diferente da raiz, apenas um arco de entrada será selecionado na solução. A restrição (2.54) garante que cada arco incluído na árvore é direcionado de um vértice com menor rótulo para um de maior. A restrição (2.56) atribui o rótulo 0 à raiz da árvore. As restrições (2.55) definem os limites dos rótulos. Há três casos possíveis que devem ser analisados quando for atribuído um rótulo para um vértice. Para cada aresta $\{i, j\}$, se:

- $y_{ij} = 1$, então $u_j \geq u_i + 1$.
- $y_{ji} = 1$, então $u_i \geq u_j + 1$.
- $y_{ij} = 0$ ou $y_{ji} = 0$, então $u_i - u_j \leq n-1$ e $u_j - u_i \leq n-1$.

Qualquer atribuição de rótulos que satisfaçam as condições acima retorna uma solução viável.

Desroches e Laporte [19] observaram que a desigualdade (2.54) pode ser fortalecida com o seguinte *lifting*:

$$u_i - u_j + n y_{ij} + (n-2) y_{ji} \leq n-1, \forall (i,j) \in A, j \neq r \qquad (2.59)$$

Gouveia [27] apresentou outras desigualdades válidas para o modelo:

$$\sum_{(k,j)\in\delta^-(j)\setminus(i,j)} y_{kj} + u_i - u_j + n y_{ij} \leq n-1, \forall (i,j) \in A, j \neq r$$
$$\sum_{(k,j)\in\delta^-(j)\setminus(i,j)} y_{kj} + u_i - u_j + n y_{ij} + (n-3) y_{ji} \leq n-1, \forall (i,j) \in A, j \neq r \quad (2.60)$$

Similarmente ao caso de fluxo simples, vale destacar a dependência da escolha da raiz r para a qualidade da relaxação linear de $\text{MST}^r_{\text{MTZ}}$, dado que pode ocorrer $P^r_{\text{MTZ}} \neq P^{r'}_{\text{MTZ}}$ quando $r \neq r'$.

De todo modo, o limite inferior gerado costuma ser inferior àquele fornecido pela formulação $\text{MST}_{\text{subtour}}$, que é igual ao valor ótimo. Em geral, se P^r_{MTZ} é projeção sobre x do conjunto viável da relaxação linear de $\text{MST}^r_{\text{MTZ}}$, temos:

Proposição 2.6 *Para todo $r \in V$, $P_{subtour} \subseteq P^r_{MTZ}$.*

2.5 Comparação Teórica entre as Relaxações Lineares

As proposições 2.1 a 2.6 são um indicativo da qualidade das formulações para MST apresentadas neste capítulo. Aqui, apresentamos mais alguns detalhes sobre as relaxações das formulações $\text{MST}_{\text{subtour}}$, $\text{MST}^r_{\text{cutsetD}}$, $\text{MST}_{\text{inter}}$, $\text{MST}^r_{\text{multiflow}}$, $\text{MST}^r_{\text{flowD}}$ e $\text{MST}^r_{\text{MTZ}}$. Não consideramos $\text{MST}_{\text{cutset}}$, $\text{MST}_{\text{multicut}}$, $\text{MST}^r_{\text{multiflowD}}$, $\text{MST}^r_{\text{flow}}$ nessa análise porque, entre as seis primeiras, há sempre uma que emprega a mesma ideia para garantir a conectividade, com um número menor de variáveis e restrições ou produzindo uma relaxação melhor.

Dada uma formulação F seja $PL(F)$ o valor da relaxação linear de F. Pelas proposições 2.1 a 2.6, concluímos as seguintes relações.

Proposição 2.7 *Para todo $r \in V$, vale que*

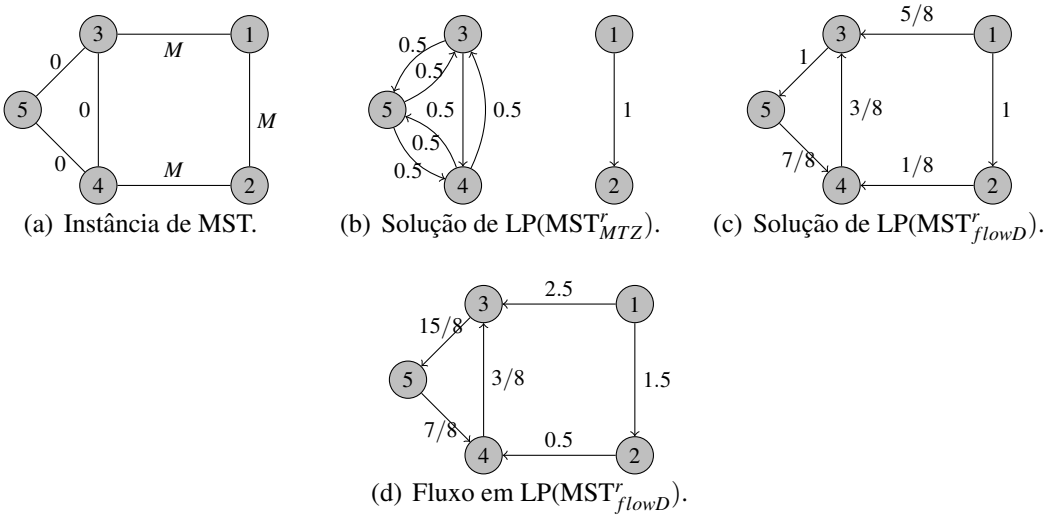
1. $PL(MST_{subtour}) = PL(MST^r_{cutsetD}) = PL(MST_{inter}) = PL(MST^r_{multiflow})$, *sendo igual ao custo de uma árvore geradora mínima;*
2. $PL(MST_{subtour}) \geq PL(MST^r_{flowD})$;
3. $PL(MST_{subtour}) \geq PL(MST^r_{MTZ})$.

Usamos um exemplo ilustrativo para mostrar que as desigualdades na Proposição 2.7 podem ser estritas e, mais ainda, os valores comparados podem ser tão distantes quanto queiramos.

Considere o grafo da Figura 2.1(a), onde as arestas têm custo 0 ou M, conforme representado. Claramente, o custo de qualquer árvore geradora mínima é $2M$, pois precisa incluir duas entre as arestas $(1,2)$, $(1,3)$ e $(1,4)$. Considere $r = 1$.

Na Figura 2.1(b), apresentamos um ponto y ótimo para a relaxação de $\text{MST}^r_{\text{MTZ}}$, sendo $u_i = 1$ para $i = 2,3,4,5$. Note que esse ponto seria viável mesmo considerando as restrições de *lifting* (2.59) e (2.60). Portanto, o valor ótimo da relaxação linear é igual a M, levando a um gap de integralidade de M. Já nas figuras 2.1(c) e 2.1(d), temos, respectivamente, o valor das variáveis y e f não-nulas no ótimo da relaxação de $\text{MST}^r_{\text{flowD}}$. Neste caso, o limite inferior é $1.75M$ e o gap de integralidade de $0.25M$. Em ambos os casos, verificamos que o gap tende a infinito quando $M \to \infty$.

Vale a mencionar aqui dois pontos em relação à análise acima. Primeiro, essa prevalência da formulação $\text{MST}^r_{\text{flowD}}$ sobre $\text{MST}^r_{\text{MTZ}}$ pode se inverter em outros exemplos. Segundo, mudando a raiz, por exemplo para o vértice 3, ambas as relaxações retornariam o valor ótimo.



(a) Instância de MST. (b) Solução de LP(MST^r_{MTZ}). (c) Solução de LP(MST^r_{flowD}).

(d) Fluxo em LP(MST^r_{flowD}).

Figura 2.1: Comparação entre relaxações.

Uma outra comparação relevante diz respeito à dimensão das formulações, que é fator importante para o tempo computacional demandado em sua resolução. Fazemos um resumo abaixo.

Formulação	# variáveis	# restrições
$\text{MST}_{\text{subtour}}$	m	exponencial
$\text{MST}^r_{\text{cutsetD}}$	$2m$	exponencial
$\text{MST}_{\text{inter}}$	$2mn+m$	$m+n^2$
$\text{MST}^r_{\text{multiflow}}$	$2mn+2m$	$2mn+n^2+2n$
$\text{MST}^r_{\text{flowD}}$	$4m$	$2n+4m$
$\text{MST}^r_{\text{MTZ}}$	$2m+n$	$n+2m$

Nessa contagem, desprezamos constantes aditivas, bem como não consideramos as variáveis x, quando podem ser eliminadas, e restrições de limite inferior ou superior das variáveis.

3 MDF-MST – ÁRVORE GERADORA MÍNIMA COM RESTRIÇÃO DE GRAU MÍNIMO E CENTRAIS E TERMINAIS FIXOS

Seja $G = (V,E)$ um grafo conexo e não direcionado. Considere que a cada aresta $e \in E$ está associado um custo $c_e \in \mathbb{R}$. Dado um inteiro positivo d_v para cada vértice $v \in V$, o problema da Árvore Geradora Mínima com Restrição de Grau Mínimo (MD-MST – *Min-Degree Minimum Spanning Tree*) consiste em encontrar uma árvore geradora mínima de G, tal que cada vértice v, na árvore, tenha grau pelo menos d_v ou seja uma folha. Na literatura, normalmente se considera $d_v = d, \forall v \in V$. Vamos chamar os vértices com grau pelo menos d de vértices centrais.

Este é um problema bastante recente. Ele foi introduzido por Almeida, Martins e Souza [3], onde foi apresentada a sua complexidade, propriedades, formulações e um algoritmo Branch-and-bound. Neste problema os vértices centrais e folhas não são definidos a priori, assim como suas quantidades.

Neste nosso trabalho, e particularmente neste capítulo, iremos abordar uma variação do MD-MST, onde os vértices centrais e folhas são definidos a priori. Vamos chamá-lo de *Problema da Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos* (MDF-MST).

Vamos considerar também neste capítulo a versão não ponderada do problema, que chamaremos *Problema da Árvore Geradora com Restrição de Grau Mínimo e Centrais e Terminais Fixos* (MDF-ST). Note a ausência do "M"da sigla "MST". Na verdade, este é um problema de decisão, que se resume a encontrar uma solução que respeite as restrições. Na Seção 3.3, vamos demonstrar que MDF-MST é NP-Difícil, provando que MDF-ST é NP-Completo para qualquer valor de $d \geq 2$. Por outro lado, na Seção 3.4, mostramos que MDF-MST é tratável por parâmetro fixo (FPT - *Fixed Parameter Tractable*), considerando como parâmetro o número de vértices centrais.

A dificuldade de solução de MDF-MST pode ser estimada já a partir da dificuldade de se encontrar uma simples solução viável, que corresponde exatamente a resolver o problema MDF-ST, isto é, resolver um problema NP-Completo. Entretanto, algumas condições necessárias ou suficientes para viabilidade, verificáveis em tempo polinomial, podem ser derivadas. Este é o assunto da Seção 3.2, que aborda também alguns casos básicos em que a própria resolução de MDF-MST é polinomial. Procurando evidenciar mais claramente a fronteira entre a NP-completude e polinomialidade do problema, apresentamos na Seção 3.5 uma extensão do estudo de complexidade, concentrando-nos na estrutura do grafo. Antes de tudo, porém, vamos definir formalmente os problemas-alvo.

3.1 Definição de MDF-MST

Nesta seção, vamos definir formalmente o Problema da Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos (MDF-MST), foco principal do nosso trabalho, assim como alguns problemas fortemente relacionados, que serão também abor-

dados, seja como referência para os nossos desenvolvimentos, seja para considerar extensões ou aplicações dos nossos resultados.

Problema 3.1 (MDF-MST) *Dado um grafo simples, conexo, $G = (C \cup T, E)$, com custo $c_e \in \mathbb{R}$, associado a cada aresta $e \in E$, e valor de grau mínimo $d_v \geq 1$, relativo a cada vértice $v \in C$, deseja-se encontrar uma árvore geradora mínima H de G, onde $d_H(v) \geq d_v$, se $v \in C$, e $d_H(v) = 1$, se $v \in T$.*

Na definição acima, $d_H(v)$ representa o grau de v na árvore H. Os vértices em C são os vértices centrais ou nós centrais, e os vértices em T são as folhas, que também iremos chamar de terminais.

Particulamente neste capítulo, vamos também considerar o problema de viabilidade associado a MDF-MST, que pode ser visto igualmente como a versão de MDF-MST com pesos unitários, uma vez que nesse caso o custo de qualquer solução viável é $n-1$. Esse será chamado Problema da Árvore Geradora com Restrição de Grau Mínimo e Centrais e Terminais Fixos (MDF-ST).

Problema 3.2 (MDF-ST) *Dado um grafo simples, conexo, $G = (C \cup T, E)$, e valor de grau mínimo $d_v \geq 1$, relativo a cada vértice $v \in C$, existe uma árvore geradora H de G, onde $d_H(v) \geq d_v$, se $v \in C$, e $d_H(v) = 1$, se $v \in T$?*

Aplicações práticas para o MDF-MST podem ocorrer nos casos em que um conjunto de nós centrais (que podem ser centros de distribuições ou dispositivos de comunicação centralizados) e um conjunto de terminais (que podem ser clientes ou dispositivos periféricos) devem se conectar, de forma que os nós centrais devem se conectar a pelo menos d outros nós centrais ou terminais e estes últimos devem se conectar a um único nó central. Tal situação pode ocorrer em projetos de redes para sistemas VLSI, estradas, redes de energia e computadores, onde se deseja uma rede mínima com nós especiais onde a informação e instalações são centralizadas ao invés de serem espalhadas.

Em conformidade com a notação geral definida na Seção 1.2, vamos adotar a seguinte notação associada a MDF-MST e MDF-ST. O grafo induzido pelos vértices centrais será denotado por $G(C)$, enquanto $\delta(C)$, $\delta(T)$ e $\delta(T:C)$ são, respectivamente, as arestas entre centrais, entre terminais e entre terminais e centrais. A quantidade de centrais, de terminais e de vértices é $c = |C|$, $t = |T|$ e $n = c + t$. Quando o grau mínimo for comum a todos os vértices centrais, usaremos a letra d para representar esse valor comum.

Ao longo de todo o texto, quando nos referirmos a MDF-MST ou MDF-ST, vamos sempre supor que $G = (C \cup T, E)$ é conexo e que o grau, em G, de cada vértice $v \in C$ é maior ou igual a d_v, pois, caso contrário, o problema seria inviável. Por simplicidade, podemos e vamos considerar que não existem arestas entre os vértices de T ($\delta(T) = \emptyset$), pois tais arestas nunca serão escolhidas para estar na solução.

Concluímos esta seção apresentando a definição formal de dois problemas relacionados a MDF-MST, que também serão tratados neste texto, principalmente nos capítulos 6 e 5.5.

Problema 3.3 (MD-MST) *Dado um grafo simples, conexo, $G = (V,E)$, com custo $c_e \in R$, associado a cada aresta $e \in E$, e valor de grau mínimo $d_v \geq 1$, relativo a cada vértice $v \in V$, deseja-se encontrar uma árvore geradora mínima H de G, onde, para todo $v \in V$, $d_H(v) \geq d_v$ ou $d_H(v) = 1$.*

Problema 3.4 (DC-MST) *Dado um grafo simples, conexo, $G = (V,E)$, com custo $c_e \in R$, associado a cada aresta $e \in E$, e valor de grau máximo $d_v \leq n-1$, relativo a cada vértice $v \in V$, deseja-se encontrar uma árvore geradora mínima H de G, onde $d_H(v) \leq d_v$, $\forall v \in V$.*

3.2 Propriedades Básicas

Almeida, Martins e Souza [3] apresentaram propriedades para o MD-MST que nos fornecem limites inferiores e superiores para a quantidade de vértices centrais e terminais. Tal resultado nos permite enunciar a seguinte condição necessária para a viabilidade do MDF-MST.

Proposição 3.5 *Suponha $d_v = d$, $\forall v \in C$. Se MDF-MST é viável então $n - \lfloor (n-2)/(d-1) \rfloor \leq t \leq n-1$ e $1 \leq c \leq \lfloor (n-2)/(d-1) \rfloor$.*

Uma condição mais abrangente para a viabilidade do MDF-MST é dada abaixo, relacionando c, t e d.

Proposição 3.6 *Se MDF-MST é viável, então $t \geq \sum_{v \in C} d_v - 2c + 2$; e se $d_v = d$ $\forall v \in C$, então $t \geq c(d-2)+2$.*

Prova. Em qualquer solução viável H do MDF-MST, temos que $\sum_{v \in V} d_H(v) = 2(c+t-1)$ e $\sum_{v \in V} d_H(v) \geq \sum_{v \in C} d_v + t$. Logo, $2(c+t-1) \geq \sum_{v \in C} d_v + t$, ou ainda, $t \geq \sum_{v \in C} d_v - 2c + 2$. A segunda parte segue diretamente da primeira. ■

Note que, ao substituirmos $c = n - t$ na inequação $t \geq c(d-2)+2$, chegamos ao resultado da Proposição 3.5. Particularmente, para o MDF-MST ser viável com $d_v = d$, $\forall v \in C$, temos como valor mínimo e máximo para t,

$$t_{min} = c(d-2)+2 \text{ e } t_{max} = n-1$$

Similarmente, o valor mínimo e máximo para c é $c_{min} = 1$ e $c_{max} = \lfloor (t-2)/(d-2) \rfloor$.

Corolário 3.7 *Suponha $d_v = d$, $\forall v \in C$. Se MDF-MST é viável, então $d \leq \left\lfloor \frac{n-2}{c} \right\rfloor + 1$. Mais ainda, se $c \geq 2$, então $d \leq \left\lfloor \frac{n}{2} \right\rfloor$.*

Prova. Substituindo $t = n - c$ em $t \geq c(d-2)+2$ chegamos ao primeiro resultado. E se $c \geq 2$ segue-se que $d \leq \left\lfloor \frac{n-2}{2} \right\rfloor + 1 = \left\lfloor \frac{n}{2} \right\rfloor$. ■

Proposição 3.8 *Suponha que $G(C)$ possui caminho hamiltoniano, $\delta(T:C) = T \times C$ e $d_v \geq 2$, $\forall v \in V$. Se $t \geq \sum_{v \in C} d_v - 2c + 2$ (ou equivalentemente $t \geq c(d-2)+2$, quando $d_v = d \ \forall v \in C$), então MDF-MST é viável.*

Prova. Seja H uma caminho hamiltoniano em $G(C)$. Temos que $\sum_{v \in C} d_H(v) = 2(c-1)$. Logo,

$$t \geq \sum_{v \in C} d_v - 2(c-1) \geq \sum_{v \in C} (d_v - d_H(v)).$$

Como, para cada $v \in C$, $d_v - d_H(v) \geq 0$, tal diferença é exatamente o que falta para completar o grau mínimo de v. Então, dado que existem todas as arestas entre T e C, deduzimos da desigualdade destacada acima que podemos formar uma árvore geradora cumprindo as restrições de grau. ■

Os resultados acima estabelecem uma condição necessária para a viabilidade de MDF-MST, qual seja $t \geq \sum_{v \in C} d_v - 2c + 2$, que pode se tornar suficiente em certos casos. Observamos que, de fato, tal condição não é suficiente em geral.

Para ilustrar essa afirmação, consideremos, por exemplo, o caso $d = 2$, quando tal condição torna-se $t \geq 2$. Se $G(C)$ não possuir um caminho hamiltoniano, o problema é inviável para $t = 2$. Esse é o caso da figura abaixo, onde descrevemos $G(C)$. Considerando $d = 2$, vamos precisar de pelo menos 4 terminais para completar o grau dos vértices em C, embora $t_{min} = 2$.

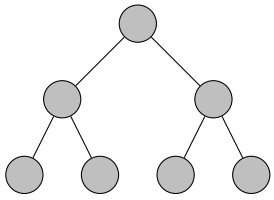


Figura 3.1: Exemplo de subgrafo G(C)

Temos que $t_{min} = c(d-2)+2 = cd-(2c-2) = cd-2(c-1)$. Analisando $cd-2(c-1)$, chegamos à conclusão que $2(c-1)$ é a contribuição, em qualquer solução viável H de MDF-MST, que as arestas de H conectando vértices de C fornecem ao grau mínimo dos vértices de C. Isto porque $H(C)$ é necessariamente uma árvore geradora de $G(C)$. Desse modo, t_{min} seria o valor mínimo necessário para gerar uma solução viável H para o problema se tivéssemos a árvore $H(C)$ com grau máximo menor ou igual a d, e os vértices com grau menor do que d ligados a uma quantidade suficiente de terminais para completar seu grau mínimo.

Como podemos ter vértice $v \in C$ com grau em $H(C)$ maior que d, ou seja, $d_{H(C)}(v) \geq d$, o número de terminais deve ser suficiente para compensar a soma desses excessos. Precisamente, devemos ter $t \geq t_{min} + \sum_{v \in C} \{\max\{0, d_{H(C)}(v) - d\}\}$. No exemplo da Figura 3.1, temos dois vértices com excesso 1, levando a $t \geq t_{min} + 2 = 4$.

Mas além de termos terminais em número suficiente, eles precisariam estar adequadamente ligados, em G, aos vértices com grau menor que d em $H(C)$. No exemplo da Figura 3.1, quatro terminais seriam suficientes apenas se existisse em G um emparelhamento entre eles e as folhas de $G(C)$.

Sendo assim, uma condição necessária e suficiente para viabilidade de MDF-MST é dada em função das árvores geradoras de $G(C)$, como segue:

Proposição 3.9 *O problema MDF-MST é viável para a instância (G,d) se, e somente se, existe uma árvore geradora H de $G(C)$ tal que o problema MDF-MST é viável para a instância $(G^{'},d)$, onde $G^{'} = (C \cup T, E^{'})$ e $E^{'} = E(H) \cup \delta(T:C)$.*

Convém então investigar o problema quando $G(C)$ é uma árvore. Neste caso, o problema de decisão MDF-ST se resume a verificar se é possível ou não ligar os vértices de T aos vértices v em $G(C)$ com grau menor que d_v, de tal forma a atingir o grau mínimo. Se for possível tal ligação, então o MDF-ST é viável; caso contrário, é inviável.

Proposição 3.10 *Se o grafo induzido $G(C)$ é uma árvore então o MDF-ST é polinomial.*

Prova. Nossa prova se resume a mostrar como podemos verificar, em tempo polinomial, se existe uma solução viável para o MDF-ST a partir de $G(C)$. Vamos reduzir nosso problema a um problema de Fluxo Máximo, de tal forma que a partir da solução do problema de Fluxo podemos concluir se existe ou não solução para o MDF-ST. Seja o grafo $G = (C \cup T, E)$ com $G(C)$ uma árvore. Vamos criar um grafo direcionado $D = (\overline{V}, A)$ com dois vértices especiais $s,t \in \overline{V}$, uma instância do problema de Fluxo Máximo. Seja $\overline{C}$ os vértices em C com grau menor que d em $G(C)$. Então fazemos $\overline{V} = \overline{C} \cup T \cup \{s,t\}$, onde s é a origem e t é o destino. Para cada aresta $(i,j) \in E$, com $i \in T$ e $j \in \overline{C}$, criamos o arco (i,j) com capacidade 1 em A. Criamos os arcos (s,i), $\forall i \in T$, com capacidade 1. Também criamos os arcos (j,t), $\forall j \in \overline{C}$, com capacidade $d_j - d(j) \geq 1$, onde $d(j)$ é o grau do vértice j na árvore $G(C)$. Ver Figura 3.2.

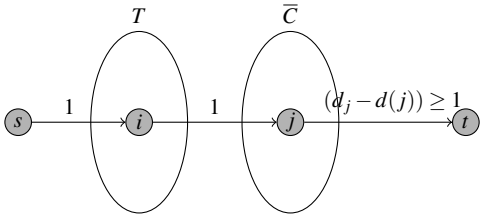


Figura 3.2: Rede D gerada a partir $G(C)$ e T para encontrar uma solução viável do MDF-MST.

Um fluxo em D define uma alocação dos vértices em T aos vértices em $\overline{C}$. Sendo assim, temos que MDF-ST é viável se, e somente se, o fluxo máximo em D, entre s e t, é $\sum_{j \in \overline{C}} (d_j - d(j))$. ■

Proposição 3.11 *Se o grafo induzido $G(C)$ é uma árvore, então MDF-MST é polinomial.*

Prova. Verificar a viabilidade de MDF-MST é polinomial pela Proposição 3.10. Se viável, resolver MDF-MST equivale a resolver um problema de fluxo de custo mínimo. Vamos criar um grafo direcionado $D = (\overline{V}, A)$ com dois vértices especiais $s,t \in \overline{V}$, uma instância do problema de Fluxo de Custo Mínimo. Então fazemos $\overline{V} = C \cup T \cup \{s,t\}$, onde s é a origem e t é o destino.

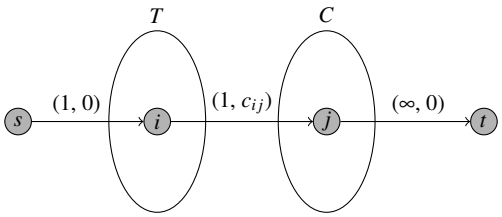


Figura 3.3: Rede D gerada a partir $G(C)$ e T para encontrar uma solução ótima do MDF-ST.

Para cada aresta $(i, j) \in E$, com $i \in T$ e $j \in C$, criamos o arco (i, j) em A, com capacidade 1 e custo igual ao custo da aresta (i, j) em G. Criamos os arcos (s, i), $\forall i \in T$, com capacidade 1 e custo 0. Também criamos os arcos (j,t), $\forall j \in C$, com capacidade ∞ e custo 0. Ver Figura 3.3.

A oferta da origem s é $b_s = |T|$. A demanda dos vértices em $u \in T$ é $b_u = 0$. A demanda dos vértices em $v \in C$ é igual a $b_v = -\max\{0, d_v - d(v)\}$, onde $d(v)$ é o grau do vértice v na árvore $G(C)$. Já a demanda do sumidouro t será dada por $b_t = -b_s - \sum_{v \in C} b_v$. O fluxo a ser enviado de s a t é $|T|$. É fácil verificar que uma solução de MDF-MST é dada por um fluxo de custo mínimo em D. ■

3.3 Complexidade de MDF-MST

Fazemos aqui uma análise da complexidade do MDF-MST, provando que se trata de um problema NP-Difícil. Para isso mostraremos que o problema não-ponderado, MDF-ST, é NP-Completo.

Reduzimos o Problema do Caminho Hamiltoniano para o MDF-ST. Um caminho hamiltoniano em um grafo é um caminho que passa por cada vértice uma única vez. O problema de decidir se um grafo tem ou não um caminho hamiltoniano entre dois vértices (H-PATH) é NP-Completo [26].

Primeiramente, consideramos o problema MDF-ST restrito a $d_v = d$ fixo (que não é parte da entrada). Iremos chamar esse problema específico de MDF-ST$_d$. Como consequência obtemos a complexidade do MDF-ST.

Teorema 3.12 *Para todo $d \geq 2$, o problema MDF-ST$_d$ (MDF-ST com d fixo) é NP-Completo.*

Prova. Primeiramente, mostramos que MDF-ST$_d$ pertence a NP. Dado um grafo $G = (V, E)$, nosso certificado é o conjunto de arestas que formam a árvore geradora. O algoritmo de verificação confere se todos os vértices de C tem grau no mínimo d, se todos os vértices de T são folhas e se o grafo é realmente uma árvore geradora. Essa verificação pode ser feita em tempo polinomial.

Agora iremos provar que H-PATH $\leq_p$ MDF-ST$_d$, o que mostra que MDF-ST$_d$ é NP-Completo. Considere uma instância do problema H-PATH, dada por um grafo $G = (V, E)$ e os vértices s e t de V. Nosso algoritmo de redução irá criar um grafo $\overline{G} = (C \cup T, \overline{E})$ que

será a instância para o MDF-ST$_d$. Começamos fazendo todos os vértices de V como vértices centrais em $\overline{G}$, ou seja, $V = C$, e induzindo as arestas de E em $\overline{E}$. Para cada vértice central $u \in C$ diferente de s e t, criamos $d-2$ novos vértices que se conectam apenas a u. Criamos ainda $d-1$ vértices que se conectam apenas a s, e $d-1$ vértices que se conectam apenas a t. Todos os vértices criados formam o conjunto T, e as arestas criadas pertencem a $\overline{E}$.

Devemos mostrar que essa transformação é uma redução, ou seja, G tem um caminho hamiltoniano entre s e t se, e somente se, o grafo $\overline{G}$ tem uma árvore geradora onde o grau de todos os vértices em C seja no mínimo d e todos os vértices em T sejam folhas. Suponha que o grafo G tenha um caminho hamiltoniano entre s e t. Esse caminho em $\overline{G}$ juntamente com vértices e arestas criados formam uma árvore geradora para $\overline{G}$ onde o grau de todos os vértices em C são d e todas as folhas estão em T. Suponha agora que o grafo $\overline{G}$ tenha uma árvore geradora H onde o grau de todos os vértices em C seja no mínimo d e todos os vértices em T sejam folhas. Como cada vértice de T tem grau 1 em $\overline{G}$, então as arestas de T para C estão na árvore H. Isso obriga que os vértices s e t se conectem a pelo menos um vértice de $C \backslash \{s,t\}$, e os outros vértices centrais devem se conectar a pelo menos dois outros vértices de C. A única forma de satisfazer estas restrições e mantendo $H(C)$ uma árvore é construindo um caminho hamiltoniano entre s e t. Concluímos que o grafo induzido $H(C)$ é um caminho hamiltoniano entre s e t. ■

Corolário 3.13 *O problema MDF-ST é NP-Completo.*

Prova. Basta considerar o caso em que $d_v = d$, $\forall v \in C$ (mas d é parte da entrada do problema). Note que d é no máximo o número de vértices. O Teorema 3.12 estabelece que, para qualquer valor de d, MDF-ST$_d$ é NP-Completo. Logo, MDF-ST é NP-Completo. ■

Como consequência do Corolário 3.13, derivamos a complexidade de MDF-MST.

Corolário 3.14 *O Problema da Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos (MDF-MST) é NP-Difícil.*

3.4 Complexidade Parametrizada

Uma forma de classificar problemas NP-Completos em escala mais detalhada encontra-se na teoria da complexidade parametrizada, onde se analisa a complexidade do problema quando um dos parâmetros a ele relacionado é fixado.

A ideia central é lidar com a intratabilidade de problemas NP-Completos, tornando-os tratáveis por parâmetro fixo, determinando condições para que a solução ótima possa ser alcançada em tempo polinomial.

A classe FPT (*Fixed-Parameter Tractable*) de problemas compreende os problemas que são tratáveis por parâmetro fixo, ou seja, cuja complexidade torna-se polinomial quando um parâmetro assume valor fixo. Mais precisamente, quando existe um algoritmo que o resolve em tempo $O(f(k)n^{\alpha})$, onde n é o tamanho da entrada não parametrizada, k o tamanho do parâmetro,

f é uma função arbitrária que define a complexidade da parte parametrizada do problema e α é uma constante independente de n e k [20],[21]. Os algoritmos para problemas FPT são chamados de algoritmos FPT. Mesmo fixando valores pequenos para os parâmetros, pode ser que esses algoritmos não sejam eficientes, por exemplo, quando $f(k) = 2^{2^{2^{2^k}}}$.

A estrutura do MDF-ST que define sua NP-Completude está relacionada à escolha da árvore geradora de $G(C)$ que nos fornece uma solução. Para cada árvore geradora, podemos verificar em tempo polinomial se, a partir dessa árvore, existe ou não uma solução viável para o problema, utilizando um algoritmo polinomial de Fluxo Máximo, conforme a prova da Proposição 3.10. Também para cada árvore geradora podemos encontrar a melhor solução de MDF-MST que tenha tal árvore como base, através de um algoritmo polinomial de Fluxo de Custo Mínimo, conforme a Proposição 3.11.

Por outro lado, enumerar todas as árvores geradoras de um grafo é um problema exponencial no tamanho do grafo. Para um grafo completo com n vértices, temos n^{n-2} árvores geradoras. Considerando, por exemplo, o algoritmo de Edmonds-Karp para o Problema de Fluxo Máximo, podemos resolver o problema MDF-ST em $O(c^{c-2}(c+t)\,|E(C,T)|^2)$.

A argumentação acima nos leva a concluir que:

Proposição 3.15 *Os problemas MDF-ST e MDF-MST estão em FPT.*

3.5 Outras Propriedades

Como visto nas seções anteriores, a complexidade de MDF-ST está fortemente ligada à estrutura do subgrafo $G(C)$. Nesta seção, procuramos avaliar melhor essa influência. Para isso, vamos nos concentrar no caso $d_v = d$, $\forall v \in C$ (hipotése considerada ao longo de toda a seção) e vamos supor que em G existem todas as arestas possíveis entre terminais e centrais, ou seja, o subgrafo induzido pelas arestas $\delta(T:C)$ é um bipartido completo. Dizemos, nesse caso, que G é T-C-completo.

Note que essa hipótese não é restritiva, no sentido de que sempre podemos acrescentar em G as arestas que faltem entre T e C e atribuir a elas custo suficientemente grande. Em outras palavras, essa hipótese está eliminando as situações de inviabilidade, atribuindo a essas situações um peso bastante grande.

Quando G é T-C-completo, a existência de uma solução de MDF-ST derivada de uma árvore geradora de $G(C)$ é dada simplesmente pela existência de um número suficiente de terminais, ou seja, um número de terminais maior ou igual à soma do que falta para completar os graus mínimos.

É fácil ver que a maior quantidade de terminais é necessária quando $G(C)$ for um grafo estrela, como ilustrado na Figura 3.4. Para esse caso será necessário ter $t \geq (d-1)(c-1) + \max\{0, d-(c-1)\}$. A segunda parte do lado direito da equação, $\max\{0, d-(c-1)\}$, é necessária pois d pode ser maior que $(c-1)$, logo $d-(c-1)$ é a quantidade necessária para tornar o grau do vértice interno da estrela viável. Definiremos então $\bar{t} = (d-1)(c-1) +$

$\max\{0, d-(c-1)\}$. Assim, podemos afirmar que, se $t \geq \bar{t}$, o problema MDF-ST se resume a encontrar uma árvore geradora H de $G(C)$, ligar cada vértice v de C com grau $d_H(v) < d$ a $d - d_H(v)$ vértices de T, e ligar cada vértice em T ainda isolado a um vértice qualquer em C.

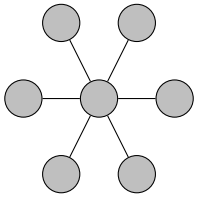


Figura 3.4: Grafo Estrela

Proposição 3.16 *Suponha que G é T-C-completo. Se $t < t_{min}$ ou $t \geq \bar{t}$, então MDF-ST é polinomial.*

Pelo resultado acima, deduzimos que o problema é polinomial se $t_{min} = \bar{t}$, pois uma das duas condições é verificada.

Comparando o valor de $\bar{t} = (d-1)(c-1) + \max\{0, d-(c-1)\}$ e $t_{min} = c(d-2) + 2$, temos,

$$t_{min} = c(d-2)+2 = (c-1+1)(d-1-1)+2 = (c-1)(d-1)+d-(c-1)$$

Como $\bar{t}$ pode assumir dois valores,

$$\bar{t} = \begin{cases} (c-1)(d-1)+d-(c-1), & \text{se } c \leq d+1 \\ (c-1)(d-1), & \text{se } c > d+1 \end{cases}$$

temos que $\bar{t} = t_{min}$ se $c \leq d+1$ e será $\bar{t} > t_{min}$ se $c > d+1$.

Logo, $t_{min} = \bar{t}$ se, e somente se, $c \leq d+1$. Pela proposição acima, obtemos a seguinte condição.

Corolário 3.17 *Suponha que G é T-C-completo. Se $c \leq d+1$ então o problema MDF-ST é polinomial.*

Uma caracterização da viabilidade de MDF-ST é apresentada abaixo.

Proposição 3.18 *Suponha que G é T-C-completo. Existe uma solução do MDF-ST se, e somente se, existe árvore geradora H de $G(C)$ tal que $\sum_{v \in P} d_H(v) \leq t - t_{min} + |P|d$, para todo $P \subseteq C$.*

Prova. ($\Longrightarrow$) (Por contra-positiva) Suponha que toda árvore geradora H de $G(C)$ possui P, subconjunto de C, tal que $p = |P|$ e $\sum_{v \in P} d_H(v) > t - t_{min} + pd$. Então,

$$\sum_{v \in C \setminus P} d_H(v) = 2(c-1) - \sum_{v \in P} d_H(v) < 2(c-1) - (t - t_{min} + pd) = (c-p)d - t$$

ou seja,

$$\sum_{v \in C \setminus P} d_H(v) + t < (c-p)d$$

Logo, mesmo usando os t terminais para aumentar o grau dos vértices de $C \setminus P$, não se poderia atingir o grau total mínimo deles, no caso, $d(c-p)$. Portanto, não existe árvore geradora de $G(C)$ com grau mínimo d.

($\Longleftarrow$) Suponha que exista árvore geradora H de $G(C)$ onde, $\forall P \subseteq C$, $\sum_{v \in P} d_H(v) \leq t - t_{min} + pd$ sendo $p = |P|$. Particularmente, esta desigualdade vale quando tomamos P como os vértices com grau maior que d na árvore geradora. Defina, $\forall v \in C \setminus P$, $\alpha_v = d - d_H(v) \geq 0$. Temos que,

$$\sum_{v \in C \setminus P} (d - \alpha_v) = \sum_{v \in C \setminus P} d_H(v) = 2(c-1) - \sum_{v \in P} d_H(v) \geq 2(c-1) + t_{min} - t - pd = (c-p)d - t$$

Logo, $\sum_{v \in C \setminus P} \alpha_v \leq t$. Isto mostra que os t terminais disponíveis são suficientes para completar o grau dos vértices em $C \setminus P$ e, consequentemente, gerar uma árvore com grau mínimo d. ■

Corolário 3.19 *Suponha que G é T-C-completo. Se $t \geq t_{min}$ e existe uma árvore geradora de $G(C)$ de grau máximo d, então existe uma solução para o MDF-ST, e uma solução é dada a partir dessa árvore geradora de grau máximo d de $G(C)$ acrescentando aos vértices de grau menor do d, arestas para os vértices de T.*

Corolário 3.20 *Suponha que G é T-C-completo e $t = t_{min}$. Existe solução para MDF-ST se, e somente se, existe árvore geradora de $G(C)$ de grau máximo menor ou igual a d.*

O corolário acima mostra que, em alguns casos, o problema MDF-ST equivale à versão de decisão de DC-MST, que é NP-Completo. Logo, temos:

Proposição 3.21 *Mesmo se G for T-C-completo, MDF-ST é NP-Completo.*

Pelo Corolário 3.19, se $t \geq t_{min}$, e $G(C)$ tem um caminho hamiltoniano então o problema é viável. Utilizando o Teorema de Dirac da teoria dos grafos que fornece uma condição suficiente para que um grafo seja hamiltoniano, chegamos a uma condição que garante a viabilidade e o tempo polinomial do problema.

Proposição 3.22 *Suponha que G é T-C-completo. Se o grafo $G(C)$ possui grau mínimo maior ou igual a $\lceil n/2 \rceil$ então $G(C)$ é hamiltoniano e MDF-ST é polinomial.*

Prova. Aplicando o Teorema de Dirac, temos que se $G(C)$ possui grau mínimo maior ou igual a $\lceil n/2 \rceil$ então $G(C)$ é hamiltoniano e para esse caso um caminho hamiltoniano pode ser obtido em tempo polinomial. Para atingir o grau mínimo d, a partir desse caminho, precisamos de apenas $(c-2)(d-2) + 2(d-1) = c(d-2) + 2 = t_{min}$ terminais. Logo, MDF-ST é polinomial. ■

Corolário 3.23 *Se G for um grafo completo então o MDF-ST é polinomial.*

Outra condição de polinomialidade é dada abaixo.

Proposição 3.24 *Suponha que G é T-C-completo. Seja $\mathscr{F}$ o conjunto de árvores geradora de $G(C)$. Seja $\Delta^* = \min\{\Delta(H) : H \in \mathscr{F}\}$. Se $d \geq \Delta^* + 1$, então MDF-ST é polinomial.*

Prova. Uma árvore geradora H de $G(C)$ com grau máximo $\Delta(H) \leq \Delta^* + 1$ pode ser encontrado em tempo polinomial [25]. Note que $\Delta(H) \leq d$. Se $t < t_{min}$, o problema é inviável. Do contrário, pelo Corolário 3.19, uma solução pode ser obtida facilmente a partir de H. ■

4 FORMULAÇÕES MATEMÁTICAS PARA MDF-MST

Neste capítulo propomos algumas formulações de programação inteira para MDF-MST, baseadas em formulações de MST apresentadas no Capítulo 2. Avaliamos teórica e computacionalmente cada modelo proposto com respeito à eficácia de sua relaxação linear, procurando um indicativo do seu potencial para a resolução efetiva do problema. Analisamos a qualidade do limite gerado e o tempo de computação requerido. Usamos instâncias da literatura relativas ao problema MD-MST, que é similar ao MDF-MST, exceto pelo fato de que os centrais e terminais não estão fixos a priori. Geramos também novas instâncias de teste, onde os custos respeitam ou não a desigualdade triangular.

4.1 Adaptações de Modelos de Árvore Geradora

Como todo problema relacionado com árvore geradora mínima, uma formulação matemática para o MDF-MST pode ser definida por qualquer conjunto de restrições que definem uma árvore geradora e pelas restrições particulares do problema. No Capítulo 2, apresentamos boa parte das formulações existentes na literatura para o MST. Aqui vamos usar as formulações $\text{MST}_{\text{subtour}}$, $\text{MST}^r_{\text{cutsetD}}$, $\text{MST}_{\text{inter}}$, $\text{MST}^r_{\text{flowD}}$ e $\text{MST}^r_{\text{MTZ}}$ como base. Tal escolha leva em conta características relevantes desses modelos ou experiências prévias bem sucedidas com problemas relacionados: os três primeiros possuem a importante e desejável propriedade da integralidade, sendo os dois primeiros as escolhas clássicas; os dois últimos destacam-se pela dimensão do modelo, que possui um número quadrático de variáveis e restrições; além disso, há relatos da eficiência de $\text{MST}_{\text{inter}}$ e $\text{MST}^r_{\text{flowD}}$ para o problema MD-MST [41], [3].

É importante lembrar aqui que uma solução viável para MDF-MST pode ser vista como uma árvore geradora (não necessariamente mínima) entre os centrais juntamente com ligações destes para o terminais, de modo a satisfazer as restrições de grau. Desse modo, nas formulações que apresentamos a seguir, aplicamos as restrições de árvore geradora apenas ao subgrafo induzido $G(C)$. Restrições próprias são empregadas para conectar os centrais com os terminais e satisfazer o grau mínimo.

Eliminação de ciclos Uma primeira formulação para o MDF-MST origina-se com o uso das restrições de eliminação de ciclos (*Subtour Elimination Constrainsts*, SECs).

$$
\begin{aligned}
(\mathrm{MDF}_{\mathrm{subtour}}) \quad & \min \sum_{e \in E} c_e x_e \\
& \text{sujeito a: } \sum_{e \in E(C)} x_e = c - 1 && (4.1)\\
& \sum_{e \in E(S)} x_e \leq |S| - 1, \forall S \subsetneq C \text{ com } |S| \geq 2 && (4.2)\\
& \sum_{e \in \delta(i)} x_e \geq d_i, \forall i \in C && (4.3)\\
& \sum_{e \in \delta(C) \cap \delta(t)} x_e = 1, \forall t \in T && (4.4)\\
& x_e \in \{0,1\}, \forall e \in E && (4.5)
\end{aligned}
$$

A função objetivo minimiza o custo da árvore geradora. As restrições (4.1)–(4.2) definem uma aŕvore geradora em $G(C)$. As restrições (4.3) garantem que cada vértice $v \in C$ tenha grau no mínimo d_v. A restrição (4.4) garante que cada vértice em T seja folha. Como visto anteriormente, podemos eliminar as arestas entre terminais, ou seja, desconsiderar as váriaveis x_e, $\forall e \in E(T)$. Note também que nas restrições de grau o corte $\delta(v)$ considerado refere-se ao grafo G e não ao subgrafo $G(C)$.

Restrições de cortes direcionados Uma segunda formulação para o MDF-MST que vamos considerar utiliza as restrições de cortes direcionados (*Directed Cutset Constraints*, DCUTs). Como no Capítulo 2, transformamos o grafo não-direcionado $G = (V,E)$ em um grafo direcionado $D = (V,A)$ e escolhemos uma raiz r. Neste caso, vamos tomar $r \in C$ e com isso eliminar, em A, os arcos de T para C, os arcos de C para r, além dos arcos entre os vértices de T.

$$
\begin{aligned}
(\mathrm{MDF}^r_{\mathrm{cutsetD}}) \quad & \min \sum_{(i,j) \in A} c_{ij} y_{ij} \\
& \text{sujeito a: } \sum_{(i,j) \in A(C)} y_{ij} = c - 1 && (4.6)\\
& \sum_{(i,j) \in \delta^+(S:C\setminus S)} y_{ij} \geq 1, \forall S \subsetneq C \text{ e } r \in S && (4.7)\\
& \sum_{(i,j) \in \delta^+(i)} y_{ij} \geq d_i - 1, \forall i \in C\setminus\{r\} && (4.8)\\
& \sum_{(i,j) \in \delta^+(r)} y_{ij} \geq d_r && (4.9)\\
& \sum_{(i,t) \in \delta^-(t) \cap \delta(C)} y_{it} = 1, \forall t \in T && (4.10)\\
& \sum_{(j,i) \in \delta^-(i) \cap A(C)} y_{ji} = 1, \forall i \in C\setminus\{r\} && (4.11)\\
& y_{ij} \in \{0,1\}, \forall (i,j) \in A && (4.12)
\end{aligned}
$$

As restrições (4.6)–(4.7) definem a árvore geradora de $G(C)$. O grau mínimo de cada vértice $i \in C$ é garantido por (4.9), se $i = r$, e por (4.8)-(4.11), se $i \neq r$. Na verdade, as restrições (4.11) são redundantes na formulação, mas fortalecem a relaxação linear. Lembre que a integralidade da formulação $\text{MST}^r_{\text{cutsetD}}$ pode ser perdida com o acréscimo das restrições de grau. Finalmente, as restrições (4.10) garantem a ligação dos terminais aos centrais, com arcos direcionados destes para aqueles.

Coincidência de arestas A formulação derivada de $\text{MST}_{\text{inter}}$, que usa a ideia de fazer coincidir n árvores, enraizadas nos diferentes vértices, é dada a seguir. Aqui podemos descrever o modelo em termos do grafo não orientado, lembrando que, para cada aresta $e \in E$, temos a variável x_e e, para cada aresta $(i, j) \in E(C)$ e cada $k \in C$, temos as variáveis λ^k_{ij} e λ^k_{ji}.

$$
\begin{aligned}
(\text{MDF}_{\text{inter}}) \quad \min & \sum_{e \in E(C)} c_e x_e \\
\text{sujeito a:} & \sum_{e \in E(C)} x_e = |C| - 1 && (4.13)\\
& x_e = \lambda^k_{ij} + \lambda^k_{ji}, \forall k \in C, \forall e = (i, j) \in E(C) && (4.14)\\
& \sum_{(i,j) \in \delta(j) \cap E(C)} \lambda^k_{ij} = 1, \forall j \in C, \forall k \in C, k \neq j && (4.15)\\
& \sum_{e \in \delta(i)} x_e \geq d_i, \forall i \in C && (4.16)\\
& \sum_{e \in \delta(t) \cap \delta(C)} x_e = 1, \forall t \in T && (4.17)\\
& \lambda^k_{ik} = 0, : \forall i \in C, \forall k \in C \cap \delta(i) && (4.18)\\
& \lambda^k_{ij}, \lambda^k_{ji} \in \{0, 1\}, \forall (i, j) \in E(C), \forall k \in C && (4.19)\\
& x_e \in \{0, 1\}, \forall e \in E && (4.20)
\end{aligned}
$$

Fluxo simples orientado A contraparte da formulação $\text{MST}^r_{\text{flowD}}$ para MDF-MST considera o grafo orientado $D = (V, A)$ e uma raiz $r \in C$. Novamente, podemos eliminar, em A, os arcos de T para C, os arcos de C para r, além dos arcos entre os vértices de T. Por simplicidade, na formulação abaixo, A denota este conjunto reduzido, que está também subentendido na notação δ^+ e δ^-.

$$(\text{MDF}^r_{\text{flowD}}) \ \min \sum_{(i,j)\in A} c_{ij}y_{ij}$$

$$\text{sujeito a:} \sum_{(j,i)\in\delta^-(i)} f_{ji} - \sum_{(i,j)\in\delta^+(i)} f_{ij} = 1, \forall i \in C\backslash\{r\} \tag{4.21}$$

$$y_{ij} \leq f_{ij} \leq (|C|-1)y_{ij}, \forall (i,j) \in A(C) \tag{4.22}$$

$$\sum_{(i,j)\in\delta^+(i)} y_{ij} \geq d_i - 1, \forall i \in C\backslash\{r\} \tag{4.23}$$

$$\sum_{(i,j)\in\delta^+(r)} y_{ij} \geq d_r \tag{4.24}$$

$$\sum_{(i,j)\in\delta^-(j)} y_{ij} = 1, \forall j \in V \setminus \{r\} \tag{4.25}$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{4.26}$$

$$f_{ij} \geq 0, \forall (i,j) \in A(C) \tag{4.27}$$

Rótulos nos vértices A próxima formulação baseia-se nas restrições Miller-Tucker-Zemlin (MTZ), que atribuem rótulos aos vértices, de modo a evitar a formação de ciclos entre os arcos escolhidos. Como na formulação $\text{MDF}^r_{\text{flowD}}$, consideramos o grafo orientado $D = (V,A)$ e uma raiz $r \in C$. Novamente, por simplicidade de notação, na formulação abaixo A, δ^+ e δ^- denotam conjuntos que não incluem os arcos de T para C, os arcos de C para r, além dos arcos entre os vértices de T.

$$(\text{MDF}^r_{\text{MTZ}}) \ \min \sum_{(i,j)\in A} c_{ij}y_{ij}$$

$$\text{sujeito a:} \sum_{(i,j)\in\delta^-(j)} y_{ij} = 1, \forall j \in V\backslash\{r\} \tag{4.28}$$

$$\sum_{(i,j)\in\delta^+(i)} y_{ij} \geq d_i - 1, \forall i \in C\backslash\{r\} \tag{4.29}$$

$$\sum_{(i,j)\in\delta^+(r)} y_{ij} \geq d_r \tag{4.30}$$

$$u_i - u_j + cy_{ij} \leq c - 1, \forall (i,j) \in A(C), j \neq r \tag{4.31}$$

$$u_i \leq c - 1, \forall i \in C\backslash\{r\} \tag{4.32}$$

$$u_i \geq 1, \forall i \in C\backslash\{r\} \tag{4.33}$$

$$u_r = 0 \tag{4.34}$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{4.35}$$

4.2 Comparação Teórica das Relaxações Lineares

Comparamos aqui os limites inferiores gerados pelas formulações $\text{MDF}_{\text{subtour}}$, $\text{MDF}_{\text{inter}}$, $\text{MDF}^r_{\text{cutsetD}}$, $\text{MDF}^r_{\text{flowD}}$ e $\text{MDF}^r_{\text{MTZ}}$. Para isso, dada uma formulação F, seja $P_x(F)$ a projeção sobre x do conjunto viável da relaxação linear de F. Para as três últimas formulações, onde as

variáveis x não aparecem explicitamente no modelo, consideramos as relações $x_e = y_{ij} + y_{ji} \in \{0,1\}$, $\forall e = (i,j) \in E$, acrescidas ao modelo para definirmos a relaxação linear. Adicionalmente, seja $PL(F)$ o valor ótimo da relaxação linear de F, ou seja, $PL(F) = \min\{\sum_{e\in E} c_e x_e : x \in P_x(F)\}$.

Proposição 4.1 *Para todo $r \in V$, vale que*

1. $PL(MDF^r_{cutsetD}) \geq PL(MDF_{subtour}) = PL(MDF_{inter})$;
2. $PL(MDF^r_{cutsetD}) \geq PL(MDF^r_{flowD})$;
3. $PL(MDF^r_{cutsetD}) \geq PL(MDF^r_{MTZ})$;

Prova. Da Proposição 2.3, temos que $P_{\text{inter}}(G(C)) = P_{\text{subtour}}(G(C))$. Além disso, as restrições de grau tanto dos centrais como dos terminais são idênticas em $\text{MDF}_{\text{subtour}}$ e $\text{MDF}_{\text{inter}}$, ambas escritas na variável x. Daí concluímos que $P_x(\text{MDF}_{\text{subtour}}) = P_x(\text{MDF}_{\text{inter}})$ e, consequentemente, temos a igualdade do item 1.

De forma análoga, das proposições 2.2, 2.5 e 2.6, para todo $r \in V$, temos que $P^r_{\text{cutsetD}}(G(C)) \subseteq P^r_{\text{flowD}}(G(C))$ e $P^r_{\text{cutsetD}}(G(C)) \subseteq P^r_{\text{MTZ}}(G(C))$. E como as restrições de grau nessas três formulações são idênticas (todas escritas na variável y), chegamos aos itens 2 e 3.

Finalmente, vamos demonstrar a desigualdade do item 1. Para isso, é suficiente mostrar que $P_x(\text{MDF}^r_{\text{cutsetD}}) \subseteq P_x(\text{MDF}_{\text{subtour}})$. Seja $x \in P_x(\text{MDF}^r_{\text{cutsetD}})$. Tome y tal que (x,y) é viável para a relaxação linear de $\text{MDF}^r_{\text{cutsetD}}$. Então

$$x_e = y_{ij} + y_{ji}, \forall e = (i,j) \in E, \quad \sum_{j\in\delta^+(r)} y_{rj} \geq d_r,$$

$$\sum_{j\in\delta^+(i)} y_{ij} \geq d_i - 1 \quad \text{e} \quad \sum_{j\in\delta^-(i)} y_{ji} = 1,$$

para todo $i \in V \setminus \{r\}$. Logo, temos que

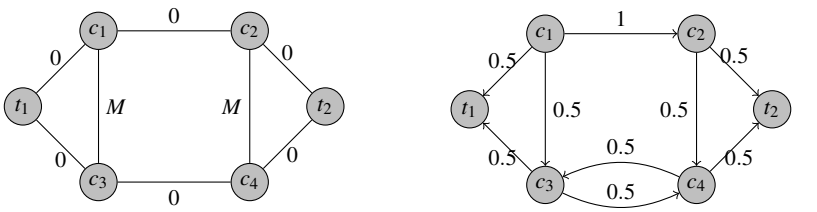
$$\sum_{e\in\delta(i)} x_e = \sum_{j\in\delta^+(i)} y_{ij} + \sum_{j\in\delta^-(i)} y_{ji} \geq d_i, \ \forall i \in V.$$

Além disso, como $x \in P^r_{\text{cutsetD}}(G(C)) = P_{\text{subtour}}(G(C))$ pela Proposição 2.2, concluímos que $x \in P_x(\text{MDF}_{\text{subtour}})$. ■

Pela proposição acima, o limite inferior gerado por $\text{MDF}^r_{\text{cutsetD}}$ é maior ou igual a dos limites gerados pelas outras formulações. Mostramos a seguir, através de um exemplo, que mesmo esse melhor limite pode ficar muito distante do ótimo.

Considere o grafo da Figura 4.1(a), onde as arestas têm custo 0 ou M, conforme indicado. Sejam c_1, c_2, c_3, c_4 os centrais e t_1, t_2 os terminais. Considere $d = 2$ e $r = c_1$. É fácil perceber que as duas arestas de peso M devem estar em qualquer solução viável, devido às restrições de grau. Sendo assim, uma solução ótima tem custo $2M$. Por outro lado, uma solução ótima relaxada para $\text{MDF}^r_{\text{cutsetD}}$ é apresentada na Figura 4.1(b). O custo dessa solução é M, de

modo que o gap de integralidade é M. Pela simetria do grafo, note que esse resultado independe da escolha da raiz.



(a) Custos das arestas (b) Solução de $\text{MDF}^r_{\text{cutsetD}}{}^r$ relaxado.

Figura 4.1: Instância de MDF-MST.

Embora os resultados acima mostrem que o limite da relaxação linear de qualquer das formulações possa estar muito distante do ótimo, a situação apresentada é pouco provável de acontecer em aplicações reais ou em instâncias aleatórias. Além disso, no exemplo apresentado, a fixação de uma das variáveis y associada às arestas de peso M leva a inviabilidade (se fixada em 0) ou à solução ótima (se fixada em 1). Dito isso, na próxima seção, avaliaremos a eficiência das formulações com instâncias da literatura.

4.3 Avaliação Computacional das Relaxações Lineares

Na seção 4.1, apresentamos várias formulações matemáticas para o MDF-MST, uma para cada tipo de restrições que definem árvore geradora: SECs, DCUTs, Fluxo, MTZ e interseção. Na seção 4.2, realizamos uma análise dos limites de programação linear fornecidos pelas formulações $\text{MDF}_{\text{subtour}}$, $\text{MDF}_{\text{inter}}$, $\text{MDF}^r_{\text{cutsetD}}$, $\text{MDF}^r_{\text{flowD}}$ e $\text{MDF}^r_{\text{MTZ}}$. Vimos que, mesmo aquelas cujas correspondentes para o MST tinham a propriedade de integralidade, agora podem não mais fornecer um bom limite inferior. Embora essa constatação seja desanimadora a príncipio, não podemos tomar o resultado dessa análise de pior caso como regra. Note, por exemplo, que para o próprio MST há formulações que não fecham o gap de integralidade, mas que podem ser efetivas computacionalmente. O mesmo acontece para vários outros problemas relacionados ao MST.

Nesta seção, fazemos uma avaliação computacional da qualidade dos limites de programação linear, bem como dos tempos gastos para obtê-los, considerando instâncias de teste da literatura. Quremos obter outros indicativos do potencial dessas formulações para a resolução do MDF-MST.

4.3.1 Algoritmo de Planos de Corte

Em algumas das formulações matemáticas para árvore geradora, temos um conjunto de restrições de tamanho exponencial. Isso acontece com as formulações baseadas em SECs (2.3) e DCUTs (2.12), seja para o caso direcionado como não direcionado. Nesses casos, resolver a formulação ou mesmo sua relaxação linear com todas as restrições se torna impraticável. Uma alternativa está em aplicar o método de planos de corte.

Um algoritmo de planos de corte funciona da seguinte maneira. Inicialmente resolvemos o problema com um pequeno subconjunto de restrições, ou seja, um problema relaxado. Se a solução ótima desse subproblema satisfaz todas as demais restrições do problema original, então ela também é uma solução ótima do problema completo. Caso tenhamos alguma restrição que seja violada por essa solução, adicionamo-la ao problema relaxado e voltamos a resolvê-lo. Esse processo iterativo continua até que nenhuma restrição seja violada pela solução corrente, que no caso será a solução ótima do problema original.

A principal dificuldade na implementação desse método encontra-se na identificação de uma ou mais restrições violadas, ou a verificação de que elas não existem, procedimento conhecido como separação de restrições. Quando o número de restrições a separar é muito grande, como no caso das SECs ou DCUTs, a separação normalmente é realizada resolvendo um problema de otimização para identificar a(s) mais violada(s).

Implementamos algoritmos de planos de corte para resolver as relaxações lineares das formulações $\text{MDF}_{\text{subtour}}$ e $\text{MDF}^{r}_{\text{cutsetD}}$. Em nossos experimentos, estamos utilizando o solver matemático CPLEX 11.1 para resolver os problemas relaxados. Implementamos dois procedimentos de separação, um para cada tipo de restrição, SECs (4.2) ou DCUTs (4.7), ambos de complexidade polinomial. Para separar as SECs, empregamos o algoritmo proposto por Padberg e Wolsey [45]. Já para separar as DCUTs, aplicamos o algoritmo de fluxo máximo ou corte mínimo.

Cada procedimento de separação pode encontrar até $(n-1)$ restrições violadas pela solução corrente, não necessariamente distintas, cada uma delas gerada a partir de um vértice. A política de escolha de quais restrições violadas deverão ser inseridas no problema é uma importante característica do algoritmo de planos de corte, que influencia em seu desempenho. Em nossas implementações, por simplicidade, optamos pelas políticas convencionais: incluir a primeira restrição violada encontrada ou incluir todas as restrições violadas encontradas na iteração corrente.

Em nossa primeira implementação do algoritmo de planos de corte para as DCUTs, constatamos que adicionar apenas a primeira restrição violada encontrada era sempre mais rápido que adicionar todas as restrições. Mesmo assim, em algumas instâncias, o algoritmo não finalizava, por questão de falta de memória. Para contornar esse problema, decidimos realizar uma política de remoção de restrições DCUTs não ativas que haviam sido adicionadas ao problema. A cada iteração, além de adicionarmos uma restrição violada, também removemos as restrições já adicionadas cujas folgas são maiores que uma certa constante ε. Verificamos uma melhora considerável no tempo gasto e uma redução significativa na quantidade de instâncias para as quais não encontrávamos a solução por falta de memória. Depois de alguns experimentos, utilizamos $\varepsilon = 0.5$.

Um último teste foi realizado para a versão que adiciona todas as restrições violadas a cada iteração e remove as restrições inativas. Nessa última implementação, em uma mesma iteração, evitamos que restrições iguais sejam incluídas no problema. Anteriormente, havíamos deixado a cargo do CPLEX utilizar suas rotinas para eliminar as restrições redundantes. Verificamos, porém, que nossa intervenção direta levou a tempos muito melhores, mesmo

comparados com a implementção que adiciona apenas uma restrição violada por iteração. Devido a isso, iremos utilizar essa última versão como nosso algoritmo de Plano de Cortes para nossos experimentos computacionais.

Na implementação do algoritmo de planos de corte para as SECs, verificamos que a política de adicionar todas as restrições violadas encontradas é muito mais rápida, comparada com a política de adicionar apenas uma e além disso, não trouxe melhorias quando evitamos que restrições iguais sejam inseridas. Verificamos também que a política de remover restrições inativas não trouxe melhorias e ainda causou problemas de convergência. Dessa forma, para as SECs não estamos removendo restrições e nem verificando se há restrições idênticas na mesma iteração.

4.3.2 Instâncias de Teste Preliminares

Para avaliarmos os limites de programação linear, resolvemos inicialmente usar as mesmas instâncias que são utilizadas na literatura para o MD-MST, já que esse problema e o MDF-MST estão fortemente relacionados. No trabalho de Martins e Souza [38], foram considerados três grupos de instâncias de testes, denominadas *CRD*, *SYM* e *ALM*, que são grafos completos com custos positivos nas arestas. Todas são definidas no plano Euclideano. As instâncias *CRD* são grafos de 30, 50, 70 e 100 vértices; *SYM* são grafos de 30, 50 e 70 vértices. Em *ALM* encontram-se os grafos maiores, com 100, 200, 300, 400, 500 vértices.

Os testes com as instâncias *CRD* e *SYM* mostram que elas são instâncias de fácil resolução (mesmo com as restrições de integralidade) pelo *solver* CPLEX. Assim, apresentamos os experimentos apenas para o grupo de instâncias *ALM*, que são as de maiores dimensões. Faremos testes considerando $d = 3$.

Para cada instância de *ALM*, vamos considerar a quantidade c de vértices centrais de acordo com a Tabela 4.1. Esses valores foram obtidos respeitando a Proposição 3.5, para garantir que exista solução viável. Nós consideramos os vértices centrais como sendo os primeiros vértices na ordem do arquivo de entrada, pois constatamos que outras escolhas para os vértices centrais não implicavam instâncias mais difíceis.

n	c
100	20, 30, 40 e 49
200	40, 50, 60, 70, 80, 90 e 99
300	60, 70, 80, 90, 100, 110, 120, 130, 140 e 149
400	60, 70, 80 , 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190 e 199
500	60, 70, 80 , 90, 100, 120, 140, 160, 180, 200, 220, 240 e 249

Tabela 4.1: Instâncias de Teste ALM

Para um mesmo valor de n, há três instâncias *ALM* originais. Então, considerando os diversos valores de c na Tabela 4.1, chegamos a um total de 147 instâncias distintas para MFD-MST, com $d = 3$.

Não estamos apresentando testes com $d > 3$ nesses experimentos preliminares. Pela Proposição 3.5, quanto maior for d, menor será a quantidade de centrais permitida. E, pelos

experimentos realizados, notamos que as instâncias se tornam mais fáceis quando a quantidade de vértices centrais decresce ou fica mais distante do máximo estabelecido pela Proposição 3.5. Por isso e devido ao bom número de instâcias testadas com $d = 3$, estamos omitindo testes para $d > 3$.

4.3.3 Resultados Computacionais

Os algoritmos deste trabalho foram implementados nas linguagens C/C++ e executados no sistema operacional Linux com um computador Intel Core i5, com 2.53 GHz e 4 GBytes de memória RAM. Em todos eles, o tempo máximo de execução que estabelecemos foi de 9000 segundos (2,5 horas). Em alguns casos, o tempo de execução foi maior que o tempo máximo estabelecido, isso acorreu pois a verificação do tempo foi inserida no loop mais externo de cada algoritmo, e por ventura quando o tempo máximo foi atingido a execução estava em algum processamento interno. Essa diferença de tempo foi exatamente o tempo necessário para a finalização desse processamento interno.

Utilizamos o Solver Matemático CPLEX, através da biblioteca Concert Technology, para resolver os modelos lineares. Decidimos escolher a versão 11.1 do CPLEX em vez da versão mais atual na época, a 12.2, porque a anterior apresentou, em todas as instâncias testadas, menores tempos computacionais.

Fizemos um comparativo do valor da relaxação linear das formulações $\text{MDF}_{\text{subtour}}$, $\text{MDF}^r_{\text{cutsetD}}$, $\text{MDF}^r_{\text{flowD}}$, $\text{MDF}^r_{\text{MTZ}}$ e $\text{MDF}_{\text{inter}}$ para as instâncias ALM. Para as formulações direcionadas, utilizamos a mesma raiz, em nosso caso o vértice de menor índice. Essa escolha simples da raiz foi feita após verificarmos, com experimentos computacionais preliminares, que a escolha da raiz não interfere tanto nos tempos de computação.

Para resolver as relaxações lineares das formulações $\text{MDF}_{\text{subtour}}$ e $\text{MDF}^r_{\text{cutsetD}}$, utilizamos os algoritmos de planos de corte mostrados na Subseção 4.3.1. Quanto às formulações $\text{MDF}^r_{\text{flowD}}$, $\text{MDF}^r_{\text{MTZ}}$ e $\text{MDF}_{\text{inter}}$, que são compactas, ou seja, possuem quantidade polinomial de restrições e variáveis, foram implementamos, via C++/Concert, diretamente no CPLEX.

Verificamos que os limites da relaxação linear para as formulações $\text{MDF}_{\text{subtour}}$, $\text{MDF}^r_{\text{cutsetD}}$ e $\text{MDF}_{\text{inter}}$ são muito fortes para essas instâncias, o que se deve ao fato de serem baseadas em restrições que descrevem completamente o politopo de árvores geradoras. Na verdade, uma grande quantidade de soluções da relaxação linear já são inteiras (em 96 das 147 instâncias testadas), e mesmo aquelas soluções que não são inteiras têm um valor muito próximo daquele da solução ótima. Para essas três formulações, todos os valores da relaxação linear encontrados para cada instância foram iguais (ressalvamos que em algumas casos não foi possível encontrar esse valor para a formulação $\text{MDF}_{\text{inter}}$ por falta de memória).

Já para as formulações $\text{MDF}^r_{\text{MTZ}}$ e $\text{MDF}^r_{\text{flowD}}$, nenhuma solução relaxada foi inteira. Os limites fornecidos por $\text{MDF}^r_{\text{flowD}}$ são quase sempre piores que aqueles de $\text{MDF}^r_{\text{MTZ}}$ e significantemente piores que os das outras três formulações. Os limites fornecidos por $\text{MDF}^r_{\text{MTZ}}$ se aproximam um pouco mais, embora ainda fiquem distantes.

Um resumo desses resultados é apresentado na Tabela 4.2. Calculamos o gap relativo pela fórmula (OPT-LP)/OPT, onde OPT e LP são, respectivamente, o valor da solução ótima do problema e da relaxação linear. Na tabela apresentamos o gap médio, o desvio padrão e o tempo médio gasto pela relaxação linear para as 147 instâncias.

Formulações	Média GAP	Desvio Padrão GAP	Média Tempo (s)
CutsetD^r	0,000219032	0,000603186	66,38
Subtour	0,000219032	0,000603186	72,997
Inter	0,000219032	0,000603186	748,66
MTZ^r	0,040363461	0,026774423	2,9
FlowD^r	0,070690925	0,022287836	52,34

Tabela 4.2: Média e desvio padrão dos GAP e média dos tempos

Os valores individuais para as instâncias são mostrados na Tabela 4.3. Por questão de espaço, apenas iremos mostrar as instâncias para as quais o valor ótimo da relaxação linear para as formulações $\text{MDF}_{\text{subtour}}$, $\text{MDF}^r_{\text{cutsetD}}$ e $\text{MDF}_{\text{inter}}$ não são inteiras. Para cada formulação temos uma coluna com o valor e o tempo gasto (em segundos). O sinal "-"marca os casos onde houve falta de memória. Na última coluna, temos o valor ótimo da instância obtido pelo algoritmo da Seção 5.1.

Já com relação ao custo computacional, verificamos que a formulação $\text{MDF}_{\text{inter}}$ requer um tempo de computação e consumo de memória bem superiores. As formulações $\text{MDF}_{\text{subtour}}$ e $\text{MDF}^r_{\text{cutsetD}}$ são similares. Percebemos que, para instâncias onde a quantidade de centrais não é o limite máximo permitido, a Formulação $\text{MDF}_{\text{subtour}}$ é mais rápida; já quando a quantidade de centrais é exatamente a quantidade máxima permitida, a formulação $\text{MDF}^r_{\text{cutsetD}}$ é melhor, exceto em 4 instâncias onde ela foi ligeiramente pior. Os tempos computacionais de $\text{MDF}^r_{\text{flowD}}$ são comparáveis àqueles dispendidos por $\text{MDF}_{\text{subtour}}$ e $\text{MDF}^r_{\text{cutsetD}}$. Os tempos demandados por $\text{MDF}^r_{\text{MTZ}}$ são os menores entre todos.

Problema	c	CutsetD	T(s)	Subtour	T(s)	MTZ	T(s)	Inter	T(s)	FlowD^r	T(s)	OPT
ALM100-1	49	5756	0,16	5756	0,32	5461,51	0,14	5756	28,52	5508,39	0,4	5777
ALM100-2	40	5617	0,2	5617	0,12	5362,84	0,06	5617	4,74	5101,24	0,16	5639
	49	5673	0,26	5673	0,24	5145,6	0,14	5673	32,7	5263,65	0,46	5676
ALM200-1	90	7054,5	5,06	7054,5	3,06	6706,66	0,74	7054,5	341,64	6414,04	4,9	7055
ALM200-2	80	7842,5	3,74	7842,5	1,64	7351,79	0,52	7842,5	528,92	6976,52	2,88	7843
	99	8328	5,84	8328	16,02	7173,2	1,04	8328	679,22	7566,5	9,78	8331
ALM200-3	80	7630	4,08	7630	1,26	7276,34	0,52	7630	371,46	6819,38	2,24	7630
	90	7480,5	6,24	7480,5	2,1	7059,14	0,86	7480,5	710,8	6670,07	3,62	7482
	99	7698	8,08	7698	7,32	7185,61	0,92	7698	1075,2	6930,33	8,64	7723
ALM300-1	90	10377,5	6,06	10377,5	2,2	10138,05	0,72	10377,5	117,04	9738,71	4,24	10378
	100	9795,5	10,72	9795,5	5,82	9501,25	0,86	9795,5	384,2	9052,58	6,06	9796
	130	9002,5	33,3	9002,5	17,46	8502,59	2,1	9002,5	1604,92	8039,64	20,02	9006
	140	8979	36,04	8979	30,06	8409,66	2,92	8979	5698	8077,14	39,48	8986
	149	9298	51,16	9298	51,8	8516,63	3,72	-	-	8469,03	58,32	9317
ALM300-2	140	8419	40,44	8419	21,3	7982,84	2,86	8419	382,03	7607,36	35,42	8420
	149	8683	42,2	8683	38,4	8162,55	4,04	-	-	7925,31	59,02	8696
ALM300-3	110	9069,5	16,52	9069,5	9,48	8775,06	1,14	9069,5	621,1	8277,23	9,84	9070
	120	9097,5	30,26	9097,5	10,58	8709,66	1,6	9097,5	1232	8173,62	12,9	9098
	130	9049,5	39,7	9049,5	13,18	8614,23	2,38	9049,5	1337,64	8170,34	26,48	9051
	149	9788,5	60,58	9788,5	65,76	8753,75	3,34	9788,5	2028,07	9066,15	56,98	9800
ALM400-1	160	10316	71,56	10316	18,64	9852,76	4,52	10316	2830,61	9541,64	50,64	10322
	180	10070,25	111,46	10070,25	75,38	9553,8	7,22	-	-	9212,12	100,56	10075
	190	10455,67	163,3	10455,67	158,12	9640,8	8,46	-	-	9682,35	171,72	10459
	199	11442,5	167,46	11442,5	493,26	9814,45	8,88	-	-	10827,16	205,7	11457
ALM400-2	100	12739	15,96	12739	2,12	12423,25	0,84	12739	621,38	11982,66	6,52	12740
	110	12361	21,46	12361	2,78	12019,44	1,16	12361	1061,32	11559,99	10,46	12362
	120	11837	34,52	11837	5,72	11494,91	1,5	11837	759,22	10999,36	13,98	11838
	130	11391	50,28	11391	9,82	11041,67	1,94	11391	1464,22	10439,9	23,32	11393
	140	11000	85,06	11000	17,3	10651,3	2,78	11000	2663	9938,55	30,06	11002
	150	10856	85,24	10856	27,78	10538	3,48	10856	2533	9815,44	38,86	10858
	160	10713	132,28	10713	26,54	10347,59	4,66	10713	2007,97	9632,6	54,18	10717
	170	10634	146,24	10634	51,16	10180,53	5,92	-	-	9529,55	67,68	10637
	180	10676	159	10676	69,42	10127,02	7,32	-	-	9529,82	97,1	10680
	190	10918,5	240,4	10918,5	129,82	10181,14	10,04	-	-	9756,58	120,08	10923
	199	11336,5	267,2	11336,5	251,98	10341,61	9,6	-	-	10389,81	213,18	11340
ALM400-3	120	11932,5	25	11932,5	8,3	11516,48	1,48	11932,5	1605,38	11044,37	13,98	11933
	140	11484,5	41,64	11484,5	24	11088,16	3,1	11484,5	3042	10624,39	33,12	11486
	150	10984	60,18	10984	35,9	10592,75	3,76	10984	672,91	10013,47	40,26	10985
	160	10725	83,74	10725	39,4	10312,23	4,54	10725	7196	9764,96	59	10726
	180	10511	126,24	10511	126	9915,2	7	-	-	9565,84	109,44	10511
	190	10742	119,38	10742	265,78	10023,99	8,52	-	-	9882,13	139,2	10746
	199	11440,5	169,92	11440,5	678,64	9957,1	10,08	-	-	10587,96	181,2	11444
ALM500-1	220	11337	461,46	11337	349,12	10724,53	13,32	-	-	10164,53	244,82	11338
	240	11393	633,54	11393	406,64	10632,07	16,98	-	-	10249,5	437,26	11401
	249	11816,12	653,62	11816,12	1146,88	10725,12	18,96	-	-	10892,96	647,44	11838
ALM500-2	220	11456	348,44	11456	226,8	11008,12	13,18	-	-	10554,63	254,72	11461
	240	11741	412,26	11741	457,02	10981,66	16,88	-	-	10848,5	449,38	11753
	249	12264,5	417,88	12264,5	1117	11037,94	19	-	-	11427,66	555,14	12266
ALM500-3	200	11091	328,24	11091	108,46	10631,57	8,5	-	-	10010,61	147,68	11096
	220	11115,5	370,06	11115,5	212,64	10583,87	13,12	-	-	10015,14	241	11127
	240	11685,5	414,16	11685,5	586,36	10799,23	16,76	-	-	10715,42	475,54	11690
	249	12666,5	539,68	12666,5	2323,32	10879,92	19,56	-	-	11710,75	633,92	12686

Tabela 4.3: Comparação dos limites de Programação Linear para instâncias de Teste ALM

4.4 Instâncias de Teste

Observamos que, além das instâncias CRD e SYM, mesmo a maioria das instâncias ALM de Martins e Souza [38], quando adaptadas para o MDF-MST, podem ser resolvidas de forma ótima pela relaxação linear de algumas formulações em tempo relativamente baixo, com o CPLEX. Por isso, resolvemos criar outras instâcias, potencialmente mais difíceis, que iremos utilizar no restante do trabalho, para avaliar os algoritmos heurísticos e exatos que desenvolvemos.

Nosso objetivo é obter instâncias que possam oferecer uma maior dificuldade em sua resolução, que demandem por algoritmos mais elaborados. Concretamente, por mais difíceis entenda-se instâncias onde a solução da relaxação linear não é inteira e para as quais o solver CPLEX necessita de um maior tempo de computação para chegar à otimalidade.

Procuramos, então, identificar alguns parâmetros que afetem fortemente à dificuldade do problema e possam ser usados na geração das instâncias. Por se tratar de um probema combinatório NP-Difícil, uma parâmetro natural é o tamanho do grafo.

Por outro lado, os resultados de complexidade obtidos no Capítulo 3 sugerem que a dificuldade de MDF-MST cresce com o número de centrais. Confirmando essa expectativa, os experimentos computacionais realizados com as formulações mostram que as instâncias se tornam mais difíceis com o aumento do número de centrais ou, mais precisamente, quando, para n fixo, a quantidade de centrais se aproxima do limite máximo permitido pela Proposição 3.5.

Verificamos também experimentalmente que, variando o valor do grau mínimo entre os vértices, isto é, não usando o mesmo d para todo vértice, temos uma maior possibilidade de a solução da relaxação linear não ser inteira. Isto ocorre mesmo com instâncias euclideanas, como as ALM.

A partir dessas constatações, geramos instâncias euclideanas e não-euclideanas, de tamanho maior que ou comparável às instâncias ALM, considerando 2 tipos de relação entre o número de centrais e terminais e 2 tipos de distribuição dos graus mínimos dos vértices. Obtemos assim 4 grupos de instâncias de cada tipo (euclideanas e não euclideanas).

Para os grupos 3 e 4, fixado um valor de c, determinamos o valor de n para satisfazer as condições da Proposição 3.6 na igualdade, ou seja, $n = c(d-1)+2$, quando $d_i = d$, $\forall i \in C$, e $n = \sum_{i \in C} d_i - c + 2$, caso contrário. Em outras palavras, nesses dois grupos, a quantidade de centrais é a máxima permitida para termos viabilidade. Esperamos assim gerar instâncias bem difíceis.

Nos grupos 1 e 2, vamos determinar o tamanho n do grafo a partir de valores específicos de c, usando a relação $n = c(d-1)+2+0.4c$, quando $d = d_i$, $\forall i \in C$, e $n = \sum_{i \in C} d_i - c + 2 + 0.4c$, caso contrário. Quer dizer, essas instâncias respeitam as condições da Proposição 3.6 com uma folga de 40% da quantidade de centrais, o que tende a torná-las de complexidade menor que as dos grupos 3 e 4.

As características descritas acima para os grupos valem tanto para as instâncias

euclideanas quanto não-euclideanas. Os valores de c serão 60, 100, 200, 300, 400, 500, 600, 700, 800 e 900.

A diferenciação entre os grupos 1 e 2 e entre os grupos 3 e 4 deve-se à distribuição dos graus. Nos grupos 1 e 3, adotamos $d_i = d = 3, \forall i \in C$. As instâncias euclideanas dos grupos 2 e 4 têm graus mínimos $d_i \in \{2,3,4\}$, distribuídos entre os centrais da seguinte forma: 25% com $d_i = 2$, 60% com $d_i = 3$ e 15% com $d_i = 4$. Isto nos leva a $\sum_{i \in C} d_i = 2.9c$. Nos grupos 2 e 4 de instâncias não-euclideanas, usamos $d_i \in \{3,4,5\}$, assim distribuídos entre os vértices centrais: 60% com $d_i = 3$, 20% com $d_i = 4$ e 20% com $d_i = 5$. Assim, temos $\sum_{i \in C} d_i = 3.6c$.

As instâncias euclideanas foram criadas pelo mesmo processo que instâncias ALM, ou seja, os vértices correspondem a coordenadas no plano euclideano, escolhidas de forma aleatória, distribuídas uniformemente em um retângulo de dimensões 480 x 640. Os custos correspondem às distâncias entre os vértices. Pelo grafo ser gerado através do mesmo processo, iremos manter a denominação ALM para essas instâncias.

Para as instâncias não euclideanas, os custos serão definidos de forma aleatória, com valores uniformemente distribuídos dentro de um mesmo intervalo $[1,1000]$. Iremos chamar essas instâncias de NEU.

Na Tabela 4.4 estão listados todos os conjuntos de instâncias, 80 no total, que serão utilizados para avaliação dos algoritmos deste trabalho. Por questão de espaço, encontram-se numa mesma tabela, em 2 blocos, as instâncias ALM e NEU. Em cada bloco, temos o nome da instância, quantidade de vértices e quantidade de centrais. Temos 40 instâncias euclideanas e 40 não euclideanas.

Instâncias ALM			Instâncias NEU		
nome	n	c	nome	n	c
ALM60-1	146	60	NEU60-1	146	60
ALM60-2	140	60	NEU60-2	182	60
ALM60-3	122	60	NEU60-3	122	60
ALM60-4	116	60	NEU60-4	158	60
ALM100-1	242	100	NEU100-1	242	100
ALM100-2	232	100	NEU100-2	302	100
ALM100-3	202	100	NEU100-3	202	100
ALM100-4	192	100	NEU100-4	262	100
ALM200-1	482	200	NEU200-1	482	200
ALM200-2	462	200	NEU200-2	602	200
ALM200-3	402	200	NEU200-3	402	200
ALM200-4	382	200	NEU200-4	522	200
ALM300-1	722	300	NEU300-1	722	300
ALM300-2	692	300	NEU300-2	902	300
ALM300-3	602	300	NEU300-3	602	300
ALM300-4	572	300	NEU300-4	782	300
ALM400-1	962	400	NEU400-1	962	400
ALM400-2	922	400	NEU400-2	1202	400
ALM400-3	802	400	NEU400-3	802	400
ALM400-4	762	400	NEU400-4	1042	400
ALM500-1	1202	500	NEU500-1	1202	500
ALM500-2	1152	500	NEU500-2	1502	500
ALM500-3	1002	500	NEU500-3	1002	500
ALM500-4	952	500	NEU500-4	1302	500
ALM600-1	1442	600	NEU600-1	1442	600
ALM600-2	1382	600	NEU600-2	1802	600
ALM600-3	1202	600	NEU600-3	1202	600
ALM600-4	1142	600	NEU600-4	1562	600
ALM700-1	1682	700	NEU700-1	1682	700
ALM700-2	1612	700	NEU700-2	2102	700
ALM700-3	1402	700	NEU700-3	1402	700
ALM700-4	1332	700	NEU700-4	1822	700
ALM800-1	1922	800	NEU800-1	1922	800
ALM800-2	1842	800	NEU800-2	2402	800
ALM800-3	1602	800	NEU800-3	1602	800
ALM800-4	1522	800	NEU800-4	2082	800
ALM900-1	2162	900	NEU900-1	2162	900
ALM900-2	2072	900	NEU900-2	2702	900
ALM900-3	1802	900	NEU900-3	1802	900
ALM900-4	1712	900	NEU900-4	2342	900

Tabela 4.4: Instâncias de teste das classes ALM e NEU

4.4.1 Avaliação Computacional das Relaxações Lineares

Resolvemos realizar a avaliação computacional das relaxações lineares para essas novas instâncias. Para isso, utilizamos o algoritmo de Plano de Corte com Cplex apresentado na Seção 4.3.1. Nas Tabelas 4.5 e 4.7, estão um resumo dos resultados. Calculamos o gap relativo pela fórmula (OPT-LP)/OPT, onde OPT e LP são, respectivamente, o valor da solução ótima

do problema e da relaxação linear. Como cada formulação resolveu uma quantidade diferente de instâncias, para realizarmos uma comparação justa, na tabelas apresentamos o gap médio, o desvio padrão e o tempo médio gasto pela relaxação linear para as instâncias que todas as formulações resolveram, que foram para as 8 primeiras instância para as instâncias ALM e NEU. Nas Tabelas 4.6 e 4.8 estão os valores individuais para as instâncias que as formulações encontraram a solução da relaxação linear. Onde encontra-se "**"significa que faltou memoria e "*"ultrapassou o tempo máximo permitido de 9000 segundos.

Ao compararmos a média dos tempos das formulação, podemos errôneamente supor que a formulação MDF^{r}_{flowD} é mais rapida, mas lembramos que esse valores foram obtidos apenas para as 9 primeiras instâncias, quando consultamos os valores para instãncias maiores, verificamos que esse fato não é verdadeiro. As formulações $MDF_{subtour}$, $MDF^{r}_{cutsetD}$ e MDF_{inter} encontram o mesmo valor de relaxação linear e melhores que os encontrados pelas formulações MDF^{r}_{MTZ} e MDF_{inter}. Quando comparamos as formulações $MDF_{subtour}$ e $MDF^{r}_{cutsetD}$, não podemos afirmar que uma é melhor que a outra, já que em alguns casos a $MDF^{r}_{cutsetD}$ obtem o melhor tempo e em outros casos é a $MDF_{subtour}$.

Formulações	Média GAP	Desvio Padrão GAP	Média Tempo (s)
CutsetD^{r}	0,000168313	0,000336921	5,18
Subtour	0,000168313	0,000336921	5,43
Inter	0,000168313	0,000336921	318,63
MTZ^{r}	0,090528201	0,046779017	0,61
FlowD^{r}	0,080952317	0,008330155	4,64

Tabela 4.5: Média e desvio padrão dos GAP e média dos tempos para as instancias ALM

Problema	CutsetD	T(s)	Subtour	T(s)	MTZ	T(s)	Inter	T(s)	FlowD^r	T(s)	w
ALM60-1	6943	1,28	6943	0,9	6490,5	0,2	6943	94,74	6360,05	0,64	6943
ALM60-2	6772	1,02	6772	0,78	6361,23	0,22	6772	16,86	6233,77	0,78	6772
ALM60-3	6709	0,94	6709	0,92	5866,28	0,22	6709	64,92	6085,31	1,08	6709
ALM60-4	7040	1	7040	0,9	5946,84	0,22	7040	110,54	6531,29	1	7040
ALM100-1	8209,5	11,36	8209,5	6,26	7887	1	8209,5	329,74	7470,12	8,3	8217
ALM100-2	8065,5	9,76	8065,5	5,32	7810,5	0,96	8065,5	395,98	7399,84	6,98	8069
ALM100-3	8055	9,4	8055	11,54	7171,58	0,88	8055	813	7451,39	10,02	8055
ALM100-4	8139	6,68	8139	16,84	7035,06	1,18	8139	723,32	7567,69	8,34	8139
ALM200-1	11509	327,68	11509	141,96	11011	9,54	**		10339,52	154,32	11509
ALM200-2	11288,25	282,18	11288,25	172,48	10722,02	8,54			10137,38	149,78	11305
ALM200-3	11433	252,32	11433	596,6	10014,72	9,78			10536,3	208,44	11454
ALM200-4	11362,5	205,18	11362,5	729,12	9836,9	9,9			10583,05	220,2	11391
ALM300-1	13457	2074	13457	2312	12813,5	35,58			12168,97	798,88	13462
ALM300-2	13002,5	2294,3	13002,5	1739	12375,17	34,46			11726,12	870	13004
ALM300-3	12779	932			11568,05	35,04			11785,79	1277,4	12797
ALM300-4	12519	698,86	12519	5863	11437,17	36,76			11606,99	1255,2	12545
ALM400-1	**		15514	8941	**				14001,38	2573	
ALM400-2	**		15107	6562					13601,81	2931,32	
ALM400-3	**		*						14543,95	4288	
ALM400-4	15239,75	3480							14381,49	4075	15255
ALM500-1	**								15488,25	8546	
ALM500-2									*		

Tabela 4.6: Comparação dos limites de Programação Linear para instâncias de Teste ALM

Formulações	Média GAP	Desvio Padrão GAP	Média Tempo (s)
CutsetD^r	0,000254353	0,00048327	5,12
Subtour	0,000254353	0,00048327	4,93
Inter	0,000254353	0,00048327	239,52
MTZ^r	0,00583772	0,005960462	0,76
FlowD^r	0,070631203	0,02686164	4,64

Tabela 4.7: Média e desvio padrão dos GAP e média dos tempos para as instancias ALM

Problema	CutsetD	T(s)	Subtour	T(s)	MTZ	T(s)	Inter	T(s)	FlowD^r	T(s)	w
NEU60-1	3032	1,26	3032	1,04	3032	0,24	3032	38,7	2769,8	0,7	3032
NEU60-2	3872	0,66	3872	0,14	3870	0,24	3872	11,64	3672,71	0,94	3872
NEU60-3	2970,5	0,92	2970,5	0,38	2941,41	0,28	2970,5	163,56	2721,22	1,16	2972
NEU60-4	4031	0,86	4031	0,4	3970,52	0,3	4031	37,64	3841,43	1,16	4031
NEU100-1	3079	13,8	3079	12,64	3079	0,88	3079	523,6	2808,23	7,6	3079
NEU100-2	3858	10,74	3858	12,5	3852	1,16	3858	260,62	3688,15	7,82	3858
NEU100-3	3083,5	8,08	3083,5	4,44	3056	1,64	3083,5	543,82	2724,11	8,02	3084
NEU100-4	3650	4,68	3650	7,9	3617,52	1,34	3650	336,62	3483,1	9,78	3655
NEU200-1	3326	527,5	3326	324,44	3326	8,96	**		3078,96	123,02	3326
NEU200-2	4280	446,84	4280	226,96	4266	14,46			4014,17	134,64	4280
NEU200-3	3386,5	226,32	3386,5	199,34	3352,04	19,26			3104,55	202,46	3388
NEU200-4	4221,56	328	4221,56	358,78	4176,04	21,84			3987,71	244,96	4223
NEU300-1	3462	5359	*		3459	44,44			3196,32	794,92	3462
NEU300-2	4382	5331	*		4379	70,42			4130,89	917	4382
NEU300-3	3323,01	2365,56	3323,01	2999,54	3295,68	92,8			3002,18	1219,94	3326
NEU300-4	4000,33	2579	*		3960,22	121,96			3724,82	1422,58	4001
NEU400-1					**				3293,72	2494	
NEU400-2									3968,51	2946,14	
NEU400-3									3184,07	4938	
NEU400-4									3924,04	5094	
NEU500-1									3245,95	6990	
NEU500-2									4130,69	6575	
NEU500-3									*		

Tabela 4.8: Comparação dos limites de Programação Linear para instâncias de Teste NEU

5 ALGORITMOS PARA MDF-MST

Neste capítulo iremos propor e descrever vários algoritmos para o MDF-MST. Desenvolvemos tanto algoritmos exatos como heurísticos, que usam ingredientes diversos. Começamos com um algoritmo de planos de corte que implementa a formulação $\text{MDF}^r_{\text{cutsetD}}$ no solver Matemático Cplex para avaliar a capacidade desse resolvedor com o MDF-MST. Apresentamos também um algoritmo de planos de corte para uma Decomposição de Benders dessa formulação. Uma terceira abordagem baseou-se em uma Relaxação Lagrangeana, usada tanto para fornecer limites inferiores (via método de subgradientes) quando superiores (via heurísticas Lagrangeanas).

Além da lagrangeana, outras heurísticas foram propostas: (i) heurísticas gulosas, que partem de uma solução que não satisfaz as restrições de grau, constroem a viabilidade, melhoram a solução com estratégias de trocas de arestas ou usam o resultado da Proposição 3.11 para obter uma solução a partir de uma árvore geradora de $G(C)$; (ii) uma busca local VNS, onde a vizinhança é definida por diferenças de arestas; (iii) um algorimo VND, que utiliza a busca VNS.

Em todos os nossos algoritmos apresentados neste Capítulo, estamos desconsiderando as arestas entre os vértices de T. Por simplicidade, em nossas exposições sobre os algoritmos, em alguns casos, iremos considerar que os valores dos graus mínimos são iguais para todos os vértices centrais.

5.1 Algoritmo de Planos de Corte com CPLEX

Como visto na Seção 4.3, usando as instâncias ALM da literatura, boa parte das soluções da relaxação linear das formulações $\text{MDF}^r_{\text{cutsetD}}$ e $\text{MDF}_{\text{subtour}}$ foram inteiras, ou seja, a solução ótima da relaxação linear também era solução ótima do problema inteiro. O tempo gasto pelo CPLEX para encontrar tal solução resolvendo essas duas formulações também foi aceitável. Aqui, avaliamos o desempenho do CPLEX para encontrar a solução ótima para as novas instâncias ALM e NEU.

Nosso algoritmo para resolver o problema inteiro também segue o método de planos de corte e é semelhante àquele usado para resolver a relaxação linear. Avaliamos duas estratégias: (i) consideramos as restrições de integralidade desde o início, resolvendo a cada iteração um subproblema com variáves inteiras, obtido daquele da iteração anterior, acrescido dos cortes encontrados ou (ii) primeiro resolvemos a relaxação linear, e só depois incluímos as restrições de integralidade, usando em ambas as etapas os planos de corte encontrados.

Depois de alguns experimentos computacionais, verificamos que a segunda estratégia parece mais promissora, mas que a política de remoção de restrições na segunda etapa não traz benefícios.

Mais precisamente, nosso algoritmo exato usando o CPLEX executa em duas etapas. Na primeira, executamos o algoritmo de planos de corte descrito na Seção 4.3, com as

mesmas políticas de geração, inserção e remoção de cortes. Caso a solução obtida não seja inteira, passamos à segunda etapa, quando convertemos as variáveis para binárias. Novamente, usamos o referido algoritmo de planos de corte, com a mesma política de inserção de cortes, mas agora não mais removendo restrições inativas.

Pelos resultados obtidos na Seção 4.3 e depois de alguns experimentos computacionais com esse algoritmo, não pudemos inferir claramente qual das duas formulações, $\text{MDF}^r_{\text{cutsetD}}$ ou $\text{MDF}_{\text{subtour}}$, é a melhor. Optamos por utilizar a formulação $\text{MDF}^r_{\text{cutsetD}}$.

5.1.1 Resultados Computacionais

Na Tabela 5.1, encontram-se os resultados do algoritmo de plano de corte usando o CPLEX para as instâncias das classes ALM e NEU. Nas colunas w_{RL} e $T_{\text{RL}}(s)$ temos o valor da relaxação linear e o tempo que o CPLEX precisou para encontrá-lo. Já nas colunas w e $T(s)$ temos o valor da solução ótima e o tempo correspondente do CPLEX.

Entradas vazias em ambas as colunas w_{RL} e $T_{\text{RL}}(s)$ significam que a relaxação linear já é inteira e que apenas a primeira fase foi necessária. Quando os quatro valores w_{RL}, $T_{\text{RL}}(s)$, w e $T(s)$ estiverem presentes, a relaxação linear não é inteira e a solução ótima foi obtida com a segunda fase do algoritmo.

No caso de os valores $T_{\text{RL}}(s)$ ou w estarem ausentes, ocorreu alguma exceção: o CPLEX acusou uma falta de memória ou ultrapassou o limite máximo de tempo de 9000 segundos. Nesse caso o valor em T(s) apresenta o tempo em que aconteceu tal exceção e, com uma simples verificação desse valor, podemos saber se houve falta de memória ou esgotou o limite de tempo. Se o valor de $T_{\text{RL}}(s)$ estiver preenchido e o de w vazio, significa que a exceção aconteceu após o cplex resolver a relaxação linear; caso contrário, foi durante. Para os casos em que aconteceu uma exceção antes de resolver a relaxação linear, o valor em w_{RL} contém o valor da solução do último subproblema resolvido, que ainda viola algumas das restrições ainda não separadas. Esse valor é um limite inferior para o problema.

O algoritmo de plano de corte foi capaz de resolver todas as instâncias das duas classes até 300 vértices centrais e uma instância ALM com $c = 400$ do grupo 4. O mesmo não aconteceu para instâncias com $c \geq 400$. Algumas instâncias pararam por ultrapassar o tempo máximo de execução e a grande maioria parou por falta de memória. Das 33 instâncias que foram resolvidas, em 12 a relaxação linear é inteira e em 21 não; e nessas 21 instâncias, o valor da relaxação linear é muito forte, bem próximo do valor da solução ótima.

Como esperado, as novas instâncias geradas trouxeram maior dificuldade ao solver CPLEX, que não foi capaz de resolver a maioria delas. Tal situação nos motivou a procurar outras alternativas de solução.

Instâncias ALM					Instâncias NEU				
nome	w_{RL}	T_{RL}(s)	w	T(s)	nome	w_{RL}	T_{RL}(s)	w	T(s)
ALM60-1			6943	1.28	NEU60-1			3032	1.26
ALM60-2			6772	1.00	NEU60-2			3872	0.66
ALM60-3			6709	0.94	NEU60-3	2970.50	0.92	2972	1.02
ALM60-4			7040	1.00	NEU60-4			4031	0.86
ALM100-1	8209.50	11.36	8217	11.96	NEU100-1			3079	13.80
ALM100-2	8065.50	9.76	8069	10.24	NEU100-2			3858	10.74
ALM100-3			8055	9.40	NEU100-3	3083.50	8.08	3084	8.72
ALM100-4			8139	6.68	NEU100-4	3650	4.68	3655	6.38
ALM200-1	11509	327.68	11509	328.68	NEU200-1			3326	527.50
ALM200-2	11288.25	282.18	11305	332.64	NEU200-2			4280	446.84
ALM200-3	11433	252.32	11454	282.82	NEU200-3	3386.50	226.32	3388	236.70
ALM200-4	11362.50	205.18	11391	648.40	NEU200-4	4221.56	328.00	4223	360.44
ALM300-1	13457	2074	13462	2092	NEU300-1	3462	5359	3462	5361
ALM300-2	13002.50	2294.30	13004	2312.20	NEU300-2	4382	5331	4382	5351
ALM300-3	12779	932	12797	1039	NEU300-3	3323.01	2365.56	3326	2686.00
ALM300-4	12519	698.86	12545	998.22	NEU300-4	4000.33	2579	4001	2891
ALM400-1	14028.24			535.44	NEU400-1	3299.16			2484.56
ALM400-2	13639.08			808.06	NEU400-2	3972			1145
ALM400-3	14578.90			631.28	NEU400-3	3187.28			530.42
ALM400-4	15239.75	3480	15255	3621	NEU400-4	3926.07			478.96
ALM500-1	15489.74			579	NEU500-1	3246.67			823.06
ALM500-2	15399.94			533.88	NEU500-2	4131.51			3315
ALM500-3	15774.13			310.66	NEU500-3	3165.06			644.30
ALM500-4	16553.35			480.30	NEU500-4	4085.97			921.44
ALM600-1	17046.88			489.02	NEU600-1	3419.56			2016.94
ALM600-2	16737.33			401.00	NEU600-2	4314.14			1790.92
ALM600-3	17871.79			550.10	NEU600-3	3176.71			1131.18
ALM600-4	18075.03			330.30	NEU600-4	4118.25			1032
ALM700-1	18454.86			471.30	NEU700-1	3554.63			1653
ALM700-2	17878.61			518.20	NEU700-2	4505			6940
ALM700-3	19476.49			505.92	NEU700-3	3376.81			620.88
ALM700-4	19118.67			373.82	NEU700-4	4318			9003
ALM800-1	19479.86			557.22	NEU800-1	3678			9003
ALM800-2	19154.36			626.86	NEU800-2	4714.74			5264
ALM800-3	22248.89			502.68	NEU800-3	3444.90			480.50
ALM800-4	24130.44			395	NEU800-4	4453.67			1693.54
ALM900-1	20535.46			542	NEU900-1	3919			7424
ALM900-2	19981.40			556.52	NEU900-2	4815			9003
ALM900-3	25195.60			615.04	NEU900-3	3648.95			3623
ALM900-4	24549.29			582.16	NEU900-4	4610			8242

Tabela 5.1: Resultado do Cplex para as instâncias das classes ALM e NEU

5.2 Algoritmos Heurísticos

Heurísticas são algoritmos que tentam obter uma solução viável de boa qualidade em tempo computacional aceitável. Na literatura encontramos os mais variados tipos de heurísticas, que vão desde heurísticas gulosas a heurísticas probabilísticas.

Da Seção 5.1 constatamos que o nosso algoritmo de plano de corte com o solver Cplex para encontrar a solução ótima para as instâncias consideradas neste trabalho não foi eficiente, mostrando a dificuldade destas instâncias. Devido a isso, implementamos alguns tipos de heurísticas para o MDF-MST com a finalidade de encontrarmos soluções de boa qualidade. Inicialmente, desenvolvemos heurísticas gulosas baseadas em trocas de arestas, depois desenvolvemos algoritmos heurísticos baseados em busca em vizinhança.

5.2.1 Heurísticas Gulosas

Quando descartamos as restrições de grau do MDF-MST, o problema resultante consiste em encontrar uma árvore geradora mínima onde os vértices em T são folhas. Repare que os vértices centrais, em C, podem ter graus quaisquer e serem até mesmo folhas, enquanto os vértices terminais, em T, serão obrigatoriamente folhas. Chamaremos esse problema de Problema da Árvore Geradora Mínima com Terminais Fixos (MST-F). Esse problema fornece um limite inferior para o MDF-MST.

Problema 5.1 (MST-F) *Dado um grafo simples, conexo, $G = (C \cup T, E)$, com custo $c_e \in \mathbb{R}$ associado a cada aresta $e \in E$, encontrar uma árvore geradora mínima H de G, onde $d_H(v) = 1$, $\forall v \in T$.*

Podemos resolver o MST-F aplicando o algoritmo Kruskal com uma pequena modificação, como sugere a proposição a seguir.

Proposição 5.2 *Seja (G,c) uma instância de MST-F, com $G = (C \cup T, E)$. Para cada $t \in T$, seja $i_t \in C$ tal que $c_{i_t t} = \min\{c_{it} : (i,t) \in \delta(C:T)\}$. Uma solução ótima para MST-F é dada por uma árvore geradora mínima para $G(C)$ junto com as arestas $\{(i_t,t) : t \in T\}$.*

Prova. Denote por $c(H)$ o custo de um subgrafo H de G.

Claramente a solução H sugerida é viável (uma árvore onde cada vértice em T tem grau 1). Suponha, por absurdo, que não é ótima. Seja H^* uma solução ótima. Primeiro, suponha que $\delta_{H^*}(C:T) \neq \delta_H(C:T)$ e seja $e = (i,j) \in \delta_{H^*}(C:T) \setminus \delta_H(C:T)$. Considere a árvore $H'^* = H^* \setminus (i,j) \cup (i_j, j)$. Temos que $c(H'^*) + c_{i_j j} - c_{ij} \leq c(H^*)$. Logo, H'^* também é ótima. Seguindo esse processo de troca, chegamos a uma solução ótima que coincide com H em todas as arestas de $\delta_H(C:T)$. Resta, então, considerar que $\delta_{H^*}(C:T) = \delta_H(C:T)$ e $H^*(C)$ não é árvore geradora mínima para $G(C)$. Mas trocando, em H^*, a subárvore $H^*(C)$ por uma árvore geradora mínima de $G(C)$, obteríamos uma árvore de custo estritamente menor que $c(H^*)$: um absurdo. Assim, chegamos ao resultado desejado. ■

5.2.1.1 Procedimentos de Viabilidade

Na solução ótima do MST-F, alguns vértices centrais podem satisfazer sua restrição de grau e outros não. Podemos tentar, a partir da solução do MST-F , encontrar uma solução de boa qualidade para o MDF-MST. Uma heurística para isso consiste em realizar trocas de arestas, tal que alguns vértices centrais i com grau estritamente maior que d_i tenham seus graus decrementados, e os vértices centrais i com grau estritamente menor que d_i tenham seus graus incrementados. Isso deve ser feito até que as restrições de grau de todos os vértices centrais sejam satisfeitas. Mas queremos efetuar essas operações da melhor forma possível, ou seja, ao realizarmos uma troca, cuidamos para que o aumento do custo da árvore resultante seja o menor possível.

Nesse intuito, desenvolvemos procedimentos que realizam trocas de arestas até que todas as restrições de grau dos vértices centrais estejam satisfeitas. Chamaremos esses procedimentos de Procedimentos de Viabilidade (PV). Um Procedimento de Viabilidade recebe como entrada uma solução viável do MST-F; sua saída será uma solução viável do MDF-MST. Na verdade, para grafos completos com quantidade de centrais que respeita a Proposição 3.6, como ocorre com nossas instâncias de teste, é garatido que o PV sempre retorna uma solução viável do MDF-MST (Ver Proposição 5.3 abaixo). Para grafos não completos, o PV pode não conseguir encontrar uma solução viável.

Seja H uma solução viável do MST-F. Uma operação de troca de arestas é definida por um par de arestas (u,v), com $u \in H$ e $v \in G \backslash H$, tal que o grafo resultante $H' = (H \cup \{v\}) \backslash \{u\}$ seja uma nova árvore geradora. A ideia do PV é que, a cada iteração, uma troca de arestas seja realizada incrementando o grau de um vértice i com grau menor que d_i e decrementando o grau de um vértice j com grau estritamente maior que d_j.

Para cada $i \in C$ tal que $d_H(i) < d_i$, classificamos as trocas de arestas que incrementam o grau de i como segue:

- **troca** $(\mathbf{u},\mathbf{v})^{\mathbf{T}}$: denota a troca de menor custo entre todas as trocas de arestas (u,v), onde $v = (i,t) \in \delta(i)$, com $t \in T$, $(i,t) \notin H$, e $u = (j,t) \in H$, com $j \in C$, $d_H(j) > d_j$. Note que $(H \cup v) \setminus u$ é uma nova árvore geradora.
- **troca** $(\mathbf{u},\mathbf{v})^{\mathbf{C}}$: denota a troca de menor custo entre todas as trocas de arestas (u,v), onde $v = (i,\ell) \in \delta(i)$, com $\ell \in C$, $(i,\ell) \notin H$, e $u = (j,\ell) \in H$, com $j \in C$, $d_H(j) > d_j$ e u pertencente ao único ciclo formado em $H \cup v$.
- **troca** $(\mathbf{u},\mathbf{v})^{<}$: a troca de menor custo entre $(u,v)^T$ e $(u,v)^C$.

Mais precisamente, um Procedimento de Viabilidade funciona da seguinte forma. A cada iteração, escolhemos um vértice $i \in C$ com $d_H(i) < d_i$ e realizamos sua troca $(u,v)^{<}$, caso ela exista, gerando uma nova árvore H. Quando todos os centrais tiverem suas restrições de grau satisfeita, o PV termina e retorna uma solução viável do MDF-MST. Caso, em algum momento, para algum central, não seja possível incrementar o grau, o PV termina sem encontrar uma solução viável.

Proposição 5.3 *Suponha que G é um grafo completo e que $t \geq \sum_{j \in C} d_j - 2c + 2$. Para toda solução viável H de MST-F e todo vértice $i \in C$ tal que $d_H(i) < d_i$, é possível realizar uma troca $(u,v)^<$. Logo, um Procedimento de Viabilidade retorna uma solução viável de MDF-MST.*

Prova. Sejam H uma solução viável de MST-F e $i \in C$ tal que $d_H(i) < d_i$. Primeiro, observe que existe $j \in C$ tal que $d_H(j) > d_j$. Do contrário, teríamos,

$$2(t+c-1) = \sum_{j \in V} d_H(j) < \sum_{v \in C} d_j + t,$$

ou seja, $t < \sum_{j \in C} d_j - 2c + 2$, contrariando a hipótese da proposição. A desigualdade estrita na expressão acima decorre de $d_H(i) < d_i$, $d_H(j) \leq d_j$, $\forall j \in C$, e $d_H(j) = 1$, $\forall j \in T$.

Tome, então, $j \in C$ tal que $d_H(j) > d_j \geq 1$. Se existe um $t \in T$ tal que $u = (j,t) \in H$, podemos realizar uma troca $(u,v)^T$, com $v = (i,t)$, posto que G é completo e $d_H(t) = 1$. Caso não exista tal t, haverá pelo menos dois centrais vizinhos de j em H e, portanto, um deles não é vizinho a i em H. Quer dizer, deve existir $\ell \in C$ tal que $v = (i,\ell) \notin H$, $u = (j,\ell) \in H$ e u pertence ao ciclo formado em $H \cup \{v\}$. Para identificar tal ℓ, considere H enraizada em i e tome ℓ como um descendente de j. Tal vertice ℓ existe, pois $d_H(j) \geq 2$, e pertence a C, porque j não se liga, em H, a qualquer terminal. Claramente, a aresta $v = (i,\ell) \in E$ não pode ocorrer na árvore H e, quando adicionada, forma um ciclo contendo u. Sendo assim, podemos realizar a uma troca $(u,v)^C$.

Finalmente, como qualquer troca $(u,v)^T$ ou $(u,v)^C$ apenas modifica os graus de i e j, aumentando o do primeiro e reduzindo o do segundo em uma unidade cada, a inviabilidade total das restrições de grau diminui em uma unidade. Posto que MDF-MST é viável pela Proposição 3.8, o PV termina por encontrar uma solução viável. ■

A partir da descrição acima, podemos obter diversos procedimentos de viabilidade, que se diferenciam pela maneira como os vértices centrais que não respeitam a sua restrição de grau são escolhidos para terem seu grau incrementado.

No primeiro Procedimento de Viabilidade, que chamaremos de Procedimento de Viabilidade 1 (PV1), a ordem em que os vértices centrais com $d_H(i) < d_i$ são selecionados é dada pela ordem crescente dos índices dos vértices, ou seja, o vértice 1 é selecionado, depois o 2 e assim sucessivamente. Quando um vértice i é selecionado, seu grau é incrementado até que sua restrição de grau seja satisfeita.

No segundo Procedimento de Viabilidade, que iremos chamar de Procedimento de Viabilidade 2 (PV2), para cada vértice i com $d_H(i) < d_i$ encontramos sua troca $(u,v)^<$. De todas as trocas $(u,v)^<$ encontradas, aquela que fornecer o menor aumento será realizada. Assim, a cada iteração do PV2, teremos o incremento do grau do vértice relacionado a troca $(u,v)^<$ escolhida. À medida que as trocas são realizadas, o vértice i que atinge o grau d_i não é mais considerado pelo PV2. O procedimento continua até que todos os vértices tenham sua restrição de grau atendida. Repare que, no PV1, o vértice i é selecionado e terá seu grau incrementado até que respeite sua restrição de grau. Já no PV2, a cada iteração, um vértice i é selecionado, de acordo com a melhor troca $(u,v)^<$. O grau do vértice i selecionado será incrementado em uma

unidade. Na iteração seguinte, um outro vértice (igual ou diferente daquele da iteração anterior) será selecionado e assim sucessivamente.

O terceiro Procedimento de Viabilidade, que iremos chamar de Procedimento de Viabilidade 3 (PV3), é baseado no PV1, sendo que a ordem em que os vértices centrais são selecionados é feita de forma aleatória. Assim, a cada iteração, o PV3 seleciona de forma randômica um vértice central i com $d_H(i) < d_i$, onde seu grau será incrementado até que sua restrição de grau seja satisfeita.

Já o quarto Procedimento de Viabilidade, que iremos chamar de Procedimento de Viabilidade 4 (PV4), é baseado no PV2, sendo que a ordem em que os vértices centrais são selecionados é aleatória. Assim, a cada iteração, o PV4 seleciona de forma randômica um vértice central i com $d_H(i) < d_i$, que terá seu grau incrementado em uma unidade. Na iteração seguinte, um novo central é escolhido, independente da escolha anterior. Da mesma forma como no PV2, à medida que as trocas são realizadas, o vértice que tiver sua restrição de grau satisfeita não é mais considerado.

Geramos assim 4 heurísticas gulosas para o MDF-MST, cada uma usando um procedimento de viabilidade diferente. Cada heurística inicia encontrando a solução ótima do MST-F.

5.2.1.2 Kruskal Modificado

Uma outra heurística gulosa que desenvolvemos baseia-se na heurística gulosa para o MD-MST, proposta por Martins e Souza [38]. A heurística gulosa é bastante similar ao algoritmo de Kruskal. Ela começa com uma floresta F formada pelos vértices do grafo, sem as arestas e, considerando as arestas na ordem não-decrescente de seus custos, iterativamente, decide incluir ou não a aresta na floresta F, até que tenhamos $n-1$ arestas e então F será uma árvore geradora do grafo. No Kruskal, a condição para que uma aresta seja incluída em F é se essa aresta não gera um ciclo. Na heurística gulosa, além da condição do Kruskal, teremos outras duas condições: (i) se a aresta não conecta um vértice de T já conectado em F e (ii) se com a adição dessa aresta ainda podemos gerar uma árvore geradora que satisfaça as restrições de grau.

Para detalharmos a condição (ii), vamos precisar definir algumas notações e parâmetros adicionais:

- uma solução parcial é definida pelo conjunto de arestas em F durante o algoritmo;
- a solução parcial tem $n-q$ componentes após a adição da q-ésima aresta, formando uma partição de V em $V_1, V_2, \ldots, V_{n-q}$;
- $d_F(i)$, $\forall i \in V$, é o grau do vértice i na solução parcial corrente;
- $2(n-q-1)$ é a soma de graus que a solução parcial corrente ainda pode receber, depois da inclusão da q-ésima aresta, sem gerar ciclos, ou seja, para gerar uma árvore;

- *need*: é a soma de graus que a solução parcial corrente necessita, após a inclusão da q-ésima aresta, para chegar a cumprir as restrições de grau mínimo e poder gerar uma solução viável do MDF-MST;

O valor de *need* é obtido considerando a inclusão da aresta em F e a partição $V_1, V_2, \ldots, V_{n-q}$:

$$need = NCV + \sum_{r=1}^{n-q} \sum_{\substack{i \in V_r : i \in C, \\ d_{F(i)} < d_i}} (d_i - d_F(i))$$

NCV é o número de componentes de F que são viáveis, ou seja, aquelas onde o grau de seus vértices $i \in C$ é maior ou igual a d_i e o grau de seus vértices em T é 1. Cada uma dessas componentes viáveis necessita apenas da adição de uma aresta para conectar um de seus vértices em C e garantir a conectividade da árvore. O segundo termo computa a soma dos graus que faltam para as outras componentes se tornarem viáveis.

Assim, a condição (ii) é dada por $need \leq 2(n-q-1)$; sendo satisfeita a aresta considerada será adicionada em F; caso contrário, será descartada. Ao final desse procedimento, se não for encontrada uma árvore, aplicamos um procedimento simples para conectar as componentes obtidas a custo mínimo e logo após aplicamos um dos Procedimentos de Viabilidade.

5.2.1.3 Busca Local Baseada em Trocas de Arestas

Ao final de cada heurística aplicamos uma busca local baseada em trocas de arestas na solução viável do MDF-MST encontrada. Serão trocas de arestas que não irão afetar as restrições de grau dos vértices centrais. Iremos selecionar duas arestas que serão removidas e duas arestas que serão adicionadas na árvore, de tal forma que ao final a solução continue sendo viável.

Seja H uma solução viável do MDF-MST e considere arestas $v, u \in H$ e arestas $w, h \notin H$. Se $c_w + c_h < c_v + c_u$, então removemos as arestas v, u de H e adicionamos as arestas w, h em H.

Iremos realizar duas buscas locais, uma baseada em arestas incidentes nos terminais e outra baseada em arestas incidentes nos centrais.

Na busca baseada em arestas incidentes nos terminais, sejam as arestas $v = (i, t_1)$ e $u = (j, t_2)$, com $i, j \in C$ e $t_1, t_2 \in T$, ambas pertecendo a uma solução viável H. Se as arestas $w = (i, t_2)$ e $h = (j, t_1)$ existirem no grafo e se $c_w + c_h < c_v + c_u$ então realizamos essa troca de arestas e obtemos uma outra solução viável com custo menor. Para todo par de arestas v e $u \in H$, incidentes nos terminais, essa busca local realiza tal verificação. Observe que, no final da troca, os terminais t_1 e t_2 continuam com grau 1 e os centrais i e j continuam com o mesmo grau que tinham em H, ou seja, continuamos com uma solução viável do MDF-MST.

Na busca baseada em arestas incidentes nos centrais, sejam as arestas $v = (i_1, j_1)$ e $u = (i_2, j_2)$, com $i_1, i_2, j_1, j_2 \in C$, ambas pertecendo a uma solução viável H. Considere os dois pares de arestas $\{w_1 = (i_1, j_2), h_1 = (i_2, j_1)\}$ e $\{w_2 = (i_1, i_2), h_2 = (j_1, j_2)\}$. Verificamos

qual desses dois pares de arestas reconecta as componentes da árvore geradas ao removermos as arestas v e u de H. Seja w e h um tal par de arestas. Se as arestas w e h existirem no grafo e se $c_w + c_h < c_v + c_u$ então realizamos essa troca de arestas e obtemos uma outra solução viável com custo menor. Para todo par de arestas v e $u \in H$, a busca local realiza essa verificação. Observe que no final da troca, os centrais i_1, i_2, j_1 e j_2 continuam com o mesmo grau que tinham em H, ou seja, continuamos com uma solução viável do MDF-MST.

5.2.1.4 Solução Ótima Quando G(C) é uma Árvore

Provamos na Proposição 3.11 que, quando o grafo $G(C)$ for uma árvore, o MDF-MST é polinomial e se resume a um problema de fluxo de custo mínimo.

Seja H uma solução viável do MDF-MST obtida por uma das heurísticas. H pode não ser a solução ótima quando consideramos o problema com apenas as arestas entre o centrais sendo $H(C)$. Ou seja, considerando a árvore $H(C)$, pode ser que as heurísticas não tenham conectado os terminais de forma ótima. Sendo assim, implementamos o algoritmo de fluxo de custo mínimo para ser aplicado ao final de cada heurística.

Este procedimento e aquele baseado em troca de arestas podem ser conjuntamente aplicados na tentativa de melhorar a solução das heurísticas. Após alguns testes computacionais, verificamos que o tempo gasto pelo algoritmo de fluxo de custo mínimo é menor quando antes aplicamos a busca local baseada em trocas de arestas. Isso se deve ao fato de que o algoritmo de fluxo de custo mínimo executa o algoritmo de Bellman-Ford para encontrar um caminho aumentante no grafo residual. Toda vez que um caminho aumentante é encontrado e o grafo residual é modificado, aplicamos novamente o Bellman-Ford. Todas as trocas de arestas realizadas pela busca local são também realizadas pelo algoritmo de fluxo de custo mínimo. No pior caso, seriam realizadas uma de cada vez, e cada troca dessa seria feita após a execução do algoritmo de Bellman Ford. O algoritmo de Bellman-Ford tem complexidade $O(n * m)$.

Dessa forma, optamos pelo seguinte processo. Ao final da execução das heurísticas, aplicamos a Busca Local Baseada em Trocas de Arestas e depois o algoritmo de Fluxo de Custo Mínimo.

5.2.2 Busca Local VND

Desenvolvemos uma segunda busca local, mais elaborada, para ser aplicada a partir de uma solução heurística. Essa busca local é baseada no procedimento heurístico VND apresentados nos seguintes trabalhos para o problema DC-MST [46],[49]. Foram usadas três estruturas de vizinhança, que iremos chamar de *melhora_N1*, *melhora_N2* e *melhora_N3*.

No *melhora_N1*, percorremos a lista de arestas da solução viável inicial e, em cada iteração, selecionamos uma aresta candidata a ser removida. Essa aresta candidata pode ser incidente a dois vértices centrais com grau maior que d, ou incidente a um vértice central com grau maior que d e a outro vértice, seja um central com grau igual a d ou uma folha. No primeiro caso, escolhemos uma aresta para ser adicionada: a aresta de menor custo, incidente a dois

vértices centrais, que reconecta as duas componentes geradas após a remoção da aresta candidata. No segundo caso, selecionamos a aresta de menor custo, incidente, de um lado, ao vértice central com grau igual d ou ao vértice folha e, de outro lado, incidente a um vértice central qualquer, que reconecta as duas componentes geradas após a remoção da aresta candidata. Nos dois casos, se o custo da aresta adicionada for menor que o custo da aresta removida, a troca é efetivada na solução corrente; caso contrário, a troca é descartada. Observemos que numa mesma execução do *melhora_N1* podemos ter mais de uma troca de arestas feita na solução corrente.

No *melhora_N2*, percorremos a lista de arestas da solução viável inicial e, em cada iteração, selecionamos uma aresta candidata a ser removida. Essa aresta candidata deve ser incidente a dois vértices centrais, sendo que um vértice deve ter grau maior que d e o outro vértice grau igual a d. Selecionamos uma aresta para ser adicionada: a aresta de menor custo, incidente a dois vértices centrais, que reconecta as duas componentes geradas após a remoção da aresta. Se o custo da aresta adicionada for menor que o custo da aresta removida, a troca é efetivada; caso contrário, a troca é descartada na solução corrente. Repare que neste caso podemos perder a viabilidade da solução. Quando isso acontecer, executamos o Procedimento de Viabilidade no vértice que perdeu a viabilidade. Da mesma forma como acontece com *melhora_N1*, podemos ter mais de uma troca de arestas feita na solução corrente.

Já no *melhora_N3*, percorremos a lista de arestas da solução viável inicial e, em cada iteração, selecionamos uma aresta candidata a ser removida. Essa aresta candidata deve ser incidente a dois vértices centrais com grau igual a d. Selecionamos uma aresta para ser adicionada: aquela de menor custo, incidente a dois vértices centrais, que reconecta as duas componentes geradas após a remoção da aresta. Se o custo da aresta adicionada for menor que o custo da aresta removida, a troca é efetivada na solução corrente; caso contrário, a troca é descartada. Da mesma forma como acontece com o *melhora_N2*, podemos perder a viabilidade da solução. Quando isso acontecer, executamos o Procedimento de Viabilidade nos vértices que perderam a viabilidade. Da mesma forma como acontece com *melhora_N1* e no *melhora_N2*, podemos ter mais de uma troca de arestas feita na solução corrente.

Utilizando essas três estruturas de vizinhança, desenvolvemos um algoritmo de Descida em Vizinhança Variável (VND). O VND é uma estratégia de busca local que, dada uma solução inicial, tenta melhorias sucessivas dentro de uma mesma estrutura de vizinhança. Caso nenhuma melhora seja encontrada, variamos a estrutura da vizinhança. Sempre que uma melhora é encontrada em uma certa vizinhança, retorna-se à vizinhança mais restrita.

O pseudocódigo do VND que propomos pode ser visto no Algoritmo 5.1.

Algoritmo 5.1: Pseudo-Código: VND

Entrada: Solução Viável T
Saída: Solução Viável T^*

```
1  T* ← T;
2  l ← 1;
3  enquanto l ≤ 3 faça
4  |   T̄ ← solução do melhora_Nl(T);
5  |   se c(T̄) < c(T*) então
6  |   |   T* ← T̄;
7  |   |   l ← 1;
8  |   senão
9  |   |   l ← l + 1;
10 |   fim se
11 fim enqto
12 retorna T*;
```

5.2.3 Algoritmo VNS

Com a finalidade de melhorar os resultados dos limites superiores encontrados pelas heurísticas propostas até aqui, desenvolvemos um algoritmo de Busca em Vizinhança Variável (VNS). VNS é uma meta-heurística que explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança e a aplicação de um método de busca local.

Dentro do VNS, costuma-se definir dois grupos de vizinhança: um para a fase de agitação, cujo objetivo é determinar uma solução viável para iniciar a etapa de busca local; e outro para a excução da busca local propriamente dita. O VNS explora vizinhanças gradativamente maiores. Focaliza a busca em torno de uma nova solução somente se um movimento de melhora é realizado.

O pseudocódigo do VNS pode ser visto no Algoritmo 5.2, onde *refinamentoTrocas*

é a busca local baseado em trocas de arestas apresentado anteriormente:

Algoritmo 5.2: Pseudo-Código: VNS

Entrada: Solução Viável T
Saída: Solução Viável T^*

```
1  T* ← T;
2  Escolha k_max, k_step e seqs;
3  para i ← 1 até seqs hacer
4      k ← 1;
5      flag ← 0;
6      enquanto k ≤ k_max faça
7          Gere a melhor solução T' ∈ N^k_VNS(T*);
8          Obtenha T̄ aplicando o VND(T');
9          se c(T̄) < c(T*) então
10             T* ← T̄;
11             k ← 1;
12             flag ← 1;
13         senão
14             k ← k + k_step;
15         fim se
16     fim enqto
17     se flag == 1 então
18         T* ← refinamentoTrocas(T*);
19     fim se
20 fin para
21 retorna T*;
```

Definimos nossa estrutura de vizinhança responsável pela agitação do VNS baseados na ideia da busca local *k-opt*, amplamente utilizada na literatura. Em particular, *1-opt*, um caso particular de *k-opt*, é utilizada para melhorar os limites superiores do problema MD-MST em Andrade et al. [7].

No *k-opt*, k arestas são removidas da solução e substituídas por outras k arestas. A partir disso, definimos as vizinhanças N^k_{VNS}, $1 \leq k \leq n-1$. Ou seja, duas soluções (árvores) são vizinhas em N^k_{VNS} se possuem $m-k$ arestas em comum. Em nosso caso, a busca na vizinhança N^1_{VNS} será realizada de forma diferente da busca nas vizinhanças N^k_{VNS}, para $k \geq 2$.

A busca em N^1_{VNS} é realizada como em *melhora_N1*, quer dizer, o mesmo tipo de trocas são avaliadas. A diferença é que, percorrida a lista de arestas candidatas, apenas uma delas será trocada, aquela que implicar no menor custo. Em outras palavras, para cada aresta candidata a ser removida, definida como em *melhora_N1*, determinamos a aresta candidata a ser adicionada, como em *melhora_N1*, mas efetivamos apenas a troca que levar ao menor custo (mesmo que seja maior que o da árvore corrente).

Já para N^k_{VNS}, escolhemos k arestas de forma aleatória para serem removidas da

solução inicial, gerando assim $k+1$ componentes conexas. Das arestas que reconectam duas componentes, escolhemos aleatoriamente uma aresta, que será adicionada se ela for incidente a um vértice central ou a um vértice terminal com grau igual a zero. Após realizarmos a remoção e adição de arestas, aplicamos um procedimento de viabilidade. Temos duas versões para o N^k_{VNS}: uma considerando como candidatas todas as arestas do grafo e outra, apenas as arestas entre vértices centrais.

5.2.4 Heurística Lagrangeana

A Relaxação Lagrangeana é um método eficiente para encontrar bons limites inferiores. Além disso, informações geradas ao longo do processo de resolução do dual lagrangeano podem ser usadas para gerar soluções viáveis. Esses procedimentos são genericamente chamados Heurísticas Lagrangeanas.

Desenvolvemos aqui uma heurística lagrangeana para o MDF-MST. Deixaremos para a Seção 5.3 os detalhes sobre a relaxação lagrangeana que propomos. Aqui, é suficiente adiantar que, em nossa relaxação lagrangeana, iremos dualizar todas as restrições de grau dos centrais. Com isso, o problema resultante, chamado de Problema Lagrangeano, é um MST-F com custos afetados pelos multiplicadores de lagrange. Como já vimos, a solução ótima para esse problema pode ser encontrada em tempo polinomial.

Como iremos mostrar na Seção 5.3, utilizamos o Método de Subgradientes [9] para resolver o Dual Lagrangeano, ou seja, o problema de encontrar os multiplicadores de lagrange que fornecem o melhor limite inferior. A cada iteração do método de subgradientes, temos a solução do problema lagrangeano para valor corrente dos multiplicadores.

Nossa heurística lagrangeana usa a solução do problema lagrangeano para gerar uma solução viável do MDF-MST, através do Procedimento de Viabilidade. A cada iteração do método de subgradientes temos uma árvore que é a solução de um MST-F. Aplicamos um Procedimento de Viabilidade nessa árvore e encontrarmos uma solução viável para o MDF-MST; depois aplicamos a Busca Local Baseada em Trocas de Arestas incidentes a Terminais e, em seguida, a buscal local VND. Ao final da execução do método de subgradientes, aplicamos o algoritmo de Fluxo de Custo Mínimo.

Implementamos três heurísticas lagrangeanas, usando o PV1. Os resultados mostrados na próxima seção correspondem à melhor heurística lagrangeana que estamos chamando de Heurística H2.

5.2.5 Resultados Computacionais

Aqui relatamos os resultados computacionais de alguns dos algoritmos heurísticos descritos nesta seção.

Com respeito às heurísticas gulosas, apresentamos os resultados para três Procedimentos de Viabilidade (PV1, PV3 e PV4) combinados com os refinamentos de solução discutidos na Subseção 5.2.1. Não estamos exibindo os resultados para o PV2, que demandou um

tempo computacional excessivo. Também não o fazemos para a heurística Kruskal modificado por ela não apresentar bons resultados. Resolvemos manter a descrição dessas heurísticas como histórico do trabalho.

Implementamos também o algoritmos VNS, utilizando como busca local o algoritmo VND aqui apresentado. Avaliamos as duas versões para o N^k_{VNS}, como explicado anteriormente. Depois de experimentos computacioanais, verificamos que o N^k_{VNS} que considera apenas arestas entre centrais obtém melhores resultados. Dessa forma, iremos considerar apenas essa versão do N^k_{VNS}.

Os parâmetros utilizados no VNS foram $seqs = 5$, $k_{max} = 0.4 * (c-1)$ e $k_{step} = 0.04 * (c-1)$. Esses parâmetros foram baseados nos resultados obtidos por Ribeiro e Souza [46]. Nosso VNS inicia com uma solução viável obtida pelo algoritmo guloso baseado no PV1, logo o resultado obtido pelo VNS será pelo menos tão bom quando o PV1. A escolha do PV1 foi feita de forma arbitrária.

Nas tabelas 5.2 e 5.3, mostramos a média e o desvio padrão do GAP relativo às instâncias para as quais conhecemos o valor do ótimo, e a média dos tempos para todas as instâncias. O GAP é calculado como $(UB - w)/w$, onde UB é o limite superior e w o valor da solução ótima.

Comparando apenas os Procedimentos de Viabilidade entre si, verificamos que não existe hegemonia de um procedimento sobre os demais, nem mesmo entre grupos de instâncias para as duas classes ALM e NEU. O mesmo acontece quando comparamos os tempos computacionais.

Já para o VNS, verificamos que, para todas as instâncias dos grupos 1 e 2, o algoritmo melhora o resultado obtido pelo PV1. O mesmo não acontece para os grupos 3 e 4, onde ele não consegue melhorar o resultado obtido pelo PV1 em qualquer instância.

Quando analisamos a heurística lagrangeana H2, verificamos que ela se mostrou a melhor heurística para toda as instâncias testadas. Isso pode ser visto pelo gap e desvio padrão relativo médio. Já os tempos computacionais são notadamente maiores para essa heurística. Isso é natural já que na heurística lagrangeana, a cada iteração do método de subgradientes, aplicamos o PV1 para encontrar uma solução viável. O processo também fornece um limite inferior, diferentemente dos outros algoritmos.

Heurísticas	Média GAP	Desvio Padrão GAP	Média Tempo (s)
PV1	0,092164629	0,071513868	331,30
PV3	0,085576963	0,067879491	334,93
PV4	0,083106412	0,059725376	332,43
VNS	0,088494450	0,074746586	1385,57
Heurística H2	0,005649678	0,009893136	4043,93

Tabela 5.2: Média e desvio padrão dos GAP e média dos tempos para as Heurísticas para as instâncias da classe ALM

Nas Tabelas 5.4 e 5.5 detalhamos os valores das soluções obtidas por cada uma das heurísticas. Os valores ótimos conhecidos das instâncias estão na coluna w. Nas demais colunas estão apresentados os resultados de cada heurística utilizando a nomenclatura apresentada aqui,

PV1, PV3, PV4, VNS e Heurística H2. Para cada instância, o melhor valor encontrado está em negrito. Os tempo computacionais encontram-se nas Tabelas 5.6 e 5.7.

Heurísticas	Média GAP	Desvio Padrão GAP	Média Tempo (s)
PV1	0,154767983	0,13027851	501,21
PV3	0,154867417	0,139075925	489,15
PV4	0,163440706	0,135298363	493,05
VNS	0,148477565	0,136253998	1709,94
Heurística H2	0,015199073	0,022948172	4195,25

Tabela 5.3: Média e desvio padrão dos GAP e média dos tempos para as Heurísticas para as instâncias da classe NEU

Instâncias	*w*	PV1	PV3	PV4	VNS	Heurística H2
ALM60-1	6943	7100	7090	7115	7042	**6943**
ALM60-2	6772	7111	7019	7064	6962	**6772**
ALM60-3	6709	7241	7313	7138	7219	**6709**
ALM60-4	7040	7537	7586	7846	7537	**7040**
ALM100-1	8217	8382	8375	8358	8305	**8217**
ALM100-2	8069	8300	8329	8295	8248	**8069**
ALM100-3	8055	9151	9014	8908	9151	**8055**
ALM100-4	8139	9517	9266	9173	9517	**8139**
ALM200-1	11509	11869	11827	11854	11829	**11509**
ALM200-2	11305	11543	11536	11692	11532	**11317**
ALM200-3	11454	13427	13527	12930	13427	**11565**
ALM200-4	11391	13422	13366	13317	13422	**11537**
ALM300-1	13462	13749	13745	13768	13733	**13464**
ALM300-2	13004	13283	13294	13292	13188	**13004**
ALM300-3	12797	14900	14045	14576	14900	**13154**
ALM300-4	12545	14736	14522	14569	14736	**12905**
ALM400-1	-	16060	16022	16088	15967	**15526**
ALM400-2	-	15595	15581	15640	15541	**15117**
ALM400-3	-	18606	18235	18114	18606	**15659**
ALM400-4	15255	18390	18607	18016	18390	**15495**
ALM500-1	-	17875	17842	17838	17830	**17328**
ALM500-2	-	17750	17887	18009	17690	**17162**
ALM500-3	-	19804	19639	19585	19804	**17360**
ALM500-4	-	21114	20685	20732	21114	**18063**
ALM600-1	-	19553	19636	19688	19525	**19012**
ALM600-2	-	19392	19348	19417	19304	**18652**
ALM600-3	-	22892	22779	22683	22892	**19865**
ALM600-4	-	23500	23654	23127	23500	**19699**
ALM700-1	-	20974	21082	21032	20956	**20417**
ALM700-2	-	20513	20463	20551	20466	**19949**
ALM700-3	-	24330	24783	24383	24330	**21894**
ALM700-4	-	24953	24381	24648	24953	**22152**
ALM800-1	-	22214	22200	22144	22156	**21618**
ALM800-2	-	22066	22051	22019	22039	**21345**
ALM800-3	-	27406	27384	27412	27406	**25365**
ALM800-4	-	30386	30530	30153	30386	**28745**
ALM900-1	-	23293	23269	23342	23260	**22856**
ALM900-2	-	22691	22808	22829	22623	**22276**
ALM900-3	-	31989	31335	31735	31989	**30285**
ALM900-4	-	31700	31476	31338	31700	**29545**

Tabela 5.4: Resultado das Heurísticas para as instâncias da classe ALM

Instâncias	Ótima	PV1	PV3	PV4	VNS	Heurística H2
NEU60-1	3032	3149	3104	3133	3084	**3032**
NEU60-2	3872	4060	4168	4129	3985	**3872**
NEU60-3	2972	3661	3503	3615	3661	**2973**
NEU60-4	4031	4889	5130	5379	4889	**4031**
NEU100-1	3079	3132	3163	3175	3124	**3079**
NEU100-2	3858	4018	4064	4064	3992	**3859**
NEU100-3	3084	3936	4392	3931	3936	**3152**
NEU100-4	3655	4453	4146	4464	4453	**3699**
NEU200-1	3326	3484	3489	3474	3406	**3326**
NEU200-2	4280	4420	4378	4394	4388	**4284**
NEU200-3	3388	4235	4371	4387	4235	**3579**
NEU200-4	4223	5837	4948	5401	5837	**4398**
NEU300-1	3462	3564	3566	3559	3531	**3464**
NEU300-2	4382	4568	4558	4534	4524	**4382**
NEU300-3	3326	4573	4661	4702	4573	**3532**
NEU300-4	4001	4929	5129	5052	4929	**4190**
NEU400-1	-	3652	3635	3641	3630	**3549**
NEU400-2	-	4372	4369	4345	4355	**4222**
NEU400-3	-	4568	4635	4921	4568	**3826**
NEU400-4	-	5212	5164	4993	5212	**4338**
NEU500-1	-	3641	3609	3611	3594	**3513**
NEU500-2	-	4525	4520	4507	4499	**4366**
NEU500-3	-	4756	4887	4561	4756	**3930**
NEU500-4	-	5978	5375	5386	5978	**4632**
NEU600-1	-	3779	3785	3757	3765	**3676**
NEU600-2	-	4709	4715	4716	4686	**4554**
NEU600-3	-	4824	4661	4973	4824	**4064**
NEU600-4	-	5545	6209	5478	5545	**4620**
NEU700-1	-	3972	3971	3956	3932	**3823**
NEU700-2	-	4906	4925	4881	4867	**4766**
NEU700-3	-	5134	5381	4987	4959	**4507**
NEU700-4	-	5741	6066	5846	5741	**5002**
NEU800-1	-	4084	4089	4068	4051	**3962**
NEU800-2	-	5103	5104	5104	5063	**4950**
NEU800-3	-	5753	5113	4986	5753	**4466**
NEU800-4	-	5854	6128	5912	5854	**5603**
NEU900-1	-	4297	4317	4296	4267	**4186**
NEU900-2	-	5197	5195	5190	5176	**5070**
NEU900-3	-	5018	5339	5016	5161	**5175**
NEU900-4	-	6171	6149	6179	6171	**5446**

Tabela 5.5: Resultado das Heurísticas para as instâncias da classe NEU

Instâncias	PV1	PV3	PV4	VNS	Heurística H2
ALM60-1	0,06	0,03	0,03	0,74	0,24
ALM60-2	0,02	0,02	0,04	0,83	0,23
ALM60-3	0,05	0,07	0,03	0,34	0,49
ALM60-4	0,04	0,03	0,03	0,35	0,77
ALM100-1	0,2	0,32	0,2	2,62	6
ALM100-2	0,11	0,11	0,16	2,3	9,96
ALM100-3	0,25	0,17	0,25	1,12	13,75
ALM100-4	0,24	0,19	0,2	1,04	15,29
ALM200-1	4,41	2,57	3,28	22,99	35,89
ALM200-2	4,13	3,75	3,13	30,25	100,05
ALM200-3	4,86	4	2,63	10,85	151,53
ALM200-4	2,92	3,45	2,07	8,72	194,23
ALM300-1	14,96	15,06	15,26	87,81	334,84
ALM300-2	13,5	10,88	14,1	121,18	258,46
ALM300-3	17,48	14,71	16,54	38,25	425,86
ALM300-4	9,84	12,84	13,99	31,68	268,48
ALM400-1	39,02	39,11	31,69	312,46	955,18
ALM400-2	44,17	38,74	40,18	384,73	827,98
ALM400-3	60,92	60,46	75,78	120,5	1658,91
ALM400-4	50,61	59,42	54,69	107,26	1737,85
ALM500-1	77,7	125,13	124,31	794,36	2054,62
ALM500-2	118,68	104,48	146,02	800,49	2493,12
ALM500-3	159,16	151,2	199,67	280,64	2109,52
ALM500-4	163,51	150,29	161,5	286,76	3601,52
ALM600-1	218,79	231,13	209,77	1559,99	7735,19
ALM600-2	247,23	207,8	248,03	1723,09	7045,84
ALM600-3	314,44	303,56	355,69	553,07	8230,73
ALM600-4	184,93	242,19	320,33	422,26	8414,54
ALM700-1	462,69	402,27	463,43	2322,86	9049,19
ALM700-2	448,3	356,33	495,09	2583,07	8986,43
ALM700-3	623,09	589,03	524,43	1855,8	9220,05
ALM700-4	515,99	531,82	553,9	1979,55	9123,36
ALM800-1	749,17	680,78	769,45	4104,85	9180,35
ALM800-2	715,16	741,97	705,24	4919,72	9227,3
ALM800-3	1232,14	1069,05	1202,98	3233,98	9503,54
ALM800-4	1060,08	976,73	1029,2	3353,85	9538,82
ALM900-1	1203,69	1247,12	1203,15	6245,09	9620,99
ALM900-2	1247,95	1190,38	1131,94	6752,54	9457,03
ALM900-3	1786,89	2065,01	1687,97	5148,43	9997,07
ALM900-4	1454,55	1764,93	1490,92	5216,32	10172

Tabela 5.6: Tempos computacionais em segundos das Heurísticas para as instâncias da classe ALM

Instâncias	PV1	PV3	PV4	VNS	Heurística H2
NEU60-1	0,03	0,03	0,05	0,71	0,68
NEU60-2	0,16	0,14	0,15	0,85	0,91
NEU60-3	0,03	0,04	0,05	0,28	2,55
NEU60-4	0,1	0,14	0,1	0,4	1,1
NEU100-1	0,37	0,33	0,34	2,94	5,32
NEU100-2	0,67	0,54	0,69	5,34	10,6
NEU100-3	0,24	0,24	0,31	1,01	7,74
NEU100-4	0,63	0,58	0,71	1,54	13,75
NEU200-1	4,21	3,98	3,59	35,87	39,93
NEU200-2	12,9	9,65	13,21	35,52	116,04
NEU200-3	3,17	4,2	3,45	9,2	74,3
NEU200-4	8,29	8,39	7,37	14,86	158,02
NEU300-1	15,79	17	21,61	117,63	476,9
NEU300-2	72,2	58,15	52,77	183,69	648,94
NEU300-3	14,11	20,56	14,96	36,67	538,12
NEU300-4	42,25	43,19	44,39	65,93	593,37
NEU400-1	72,48	70,07	57,11	388,5	1330
NEU400-2	154,96	149,66	175,39	468,54	2290,31
NEU400-3	55,06	55,32	56,13	114,05	1320,19
NEU400-4	148,12	119,32	119,75	209,38	1738,5
NEU500-1	138,1	147,39	151,1	743,86	4188,37
NEU500-2	420,07	332,36	373,51	1104,17	3405,68
NEU500-3	153,79	118,51	123,21	278,39	3998,53
NEU500-4	311,02	332,65	335,98	443,33	3282,61
NEU600-1	285,02	323,91	293,16	1900,52	5851,4
NEU600-2	692,55	660,98	519,1	2443,96	6899,62
NEU600-3	249,86	237,96	235,89	483,82	4854,18
NEU600-4	563,27	551,46	519,78	806,45	9225,2
NEU700-1	517,56	471,11	520,91	2948,87	9180,44
NEU700-2	1332,4	1379,68	1396,04	3351,92	9337,32
NEU700-3	459,76	405,44	453,17	2226,73	9147,93
NEU700-4	1085,62	1108,16	855,08	2566,41	9482,83
NEU800-1	806,98	746,23	803,24	4753,12	9205,33
NEU800-2	2257,66	2098,15	1990,13	6401,2	10142,78
NEU800-3	679,73	709,19	672,1	3684,29	9326,72
NEU800-4	1799,73	1793,21	1693,22	4254,38	9768,61
NEU900-1	1111,5	1159,19	1450,31	6734,51	9843,66
NEU900-2	2934,8	2715,2	3393,55	9167,32	11264,88
NEU900-3	1249,76	1078,63	803,62	6147,17	9490,93
NEU900-4	2393,63	2635,1	2566,89	6264,19	10545,63

Tabela 5.7: Tempos computacionais em segundos das Heurísticas para as instâncias da classe NEU

5.3 Relaxação Lagrangeana

Em otimização combinatória, observa-se que muitos problemas considerados difíceis ou mesmo problemas cuja resolução da relaxação linear e consequente obtenção de limites inferiores demanda considerável esforço computacional podem ser a composição de problemas fáceis ou melhor tratáveis com um conjunto de restrições que os tornam complicados. Em outros termos, quando se retira esse conjunto de restrições, a resolução do problema se torna fácil e até mesmo trivial.

A dualização do conjunto de restrições complicadoras, isto é, a transferência destas para a função objetivo penalizadas por multiplicadores, chamados de Multiplicadores de Lagrange, produz um problema de fácil resolução, cujo valor da solução ótima é um limite inferior (para problemas de minimização) para o valor ótimo do problema original. Este problema é chamado de Problema Lagrangeano, e esta ideia é chamada de Relaxação Lagrangeana.

O problema lagrangeano pode, portanto, ser usado no lugar da relaxação linear para produzir limites inferiores em algoritmos do tipo *branch-and-bound*. Além disso, com base nesse limite inferior, é possível estimar quão próxima está da solução ótima uma solução viável disponível.

A relaxação lagrangeana foi desenvolvida em meados dos anos 70, no trabalho pioneiro de Held e Karp [31, 32], para o problema do caixeiro viajante, e hoje é uma técnica considerada indispensável para gerar limites inferiores para serem usadas em algoritmos de otimização combinatória.

Pensando nisso, utilizamos a ideia da relaxação lagrangeana para obter outro limite inferior e uma heurística lagrangeana para o MDF-MST. Em nosso caso, as restrições complicadoras são as que asseguram os graus mínimos dos centrais. Dualizando tais restrições em qualquer uma das formulações que propusemos para o MDF-MST, obtemos um MST-F com os custos das arestas modificados pelos multiplicadores de lagrange.

É importante destacar que, relaxando tais restrições, não esperamos obter limite de qualidade superior àquele produzido pela relaxação linear, visto que o problema lagrangeano é um MST-F. Por outro lado, vemos na relaxação lagrangeana dois outros benefícios: a possibilidade de obtenção de um limite inferior de boa qualidade em tempo menor que aquele demandado pela resolução da relaxação linear (que se mostrou muito alto, conforme Tabela 5.1) e a perspectiva de obtenção de bons limites superiores via heurísticas lagrangeanas.

Por facilidade de apresentação, vamos usar a formulação $\text{MDF}_{\text{subtour}}$, definida com as restrições de eliminação de ciclos (SECs). O desenvolvimento para as outras formulações é similar.

5.3.1 Dual Lagrangeano

Considere o problema dado por $(4.1) - (4.5)$ e o conjunto de restrições (4.3), que serão relaxadas através de multiplicadores lagrangeanos $u_i \geq 0$, $i \in C$. A função lagrangeana

assim obtida é:

$$\begin{aligned}\psi(x,u) &= \sum_{e\in E} c_e x_e + (\sum_{i\in C} u_i(d_i - \sum_{e\in\delta(i)} x_e)) \\ &= \sum_{e\in E} c_e x_e - \sum_{i\in C}\sum_{e\in\delta(i)} u_i x_e + \sum_{i\in C} u_i d_i\end{aligned}$$

Podemos decompor

$$\sum_{i\in C}\sum_{e\in\delta(i)} u_i x_e = \sum_{e=(i,j)\in E(C)} x_e(u_i+u_j) + \sum_{e=(i,t)\in E(C:T)} x_e u_i.$$

Então, temos que

$$\begin{aligned}\psi(x,u) &= \sum_{e\in E} c_e x_e - (\sum_{e=(i,j)\in E(C)} x_e(u_i+u_j) + \sum_{e=(i,t)\in E(C:T)} x_e u_i) + \sum_{i\in C} u_i d_i \\ &= \sum_{e=(i,j)\in E(C)} x_e(c_e-u_i-u_j) + \sum_{e=(i,t)\in E(C:T)} x_e(c_e-u_i) + \sum_{i\in C} u_i d_i.\end{aligned}$$

Dado $u \geq 0$, definimos a seguinte relaxação lagrangeana da formulação $\text{MDF}_{\text{subtour}}$:

$$\begin{aligned}&(RL) \quad LP(u) = \min_x \psi(x,u) \\ &\qquad\qquad \text{sujeito a: (4.1),(4.2),(4.4) e (4.5)}\end{aligned}$$

Os pontos x que satisfazem as restrições (4.1), (4.2), (4.4) e (4.5) são os vetores de incidência de árvores geradoras de $G = (C \cup T, E)$ onde os vértices em T são folhas. Sendo assim, para cada $u \geq 0$, o problema acima é um MST-F, que pode ser resolvido em tempo polinomial, através de um algorimo de árvore geradora mínima no subgrafo $G(C)$ com custos $\bar{c}_e = c_e - u_i - u_j$, $\forall e = (i,j) \in E(C)$, seguido de uma alocação ótima dos terminais aos centrais, considerando agora os custos $\bar{c}_e = c_e - u_i$, $\forall e = (i,t) \in \delta(C:T)$. Logo, $LP(u)$ pode ser computado facilmente.

Sabemos que $LP(u)$, para qualquer $u \geq 0$, é um limite inferior para o MDF-MST. O vetor u^* que fornece o melhor limite inferior $LP(u^*)$ é obtido pela resolução do seguinte dual lagrangeano:

$$(DL) \quad LP(u^*) = \max_{u\geq 0} LP(u)$$

Podemos mostrar que $LP(u)$ é uma função côncava linear por partes e que, se uma solução para $LP(u)$ ocorre em $\bar{x}$, ou seja, $LP(u) = \psi(\bar{x},u)$, então um subgradiente de $LP()$ em u é dado pelo vetor g, cuja componente $g_i = d_i - \sum_{e\in\delta(i)} \bar{x}_e$ [51].

5.3.2 Resolvendo o Dual Lagrangeano

Para encontrar o u^* que fornece o melhor valor para $LP(u)$ podemos empregar alguns métodos existentes na literatura, como o Método de Subgradientes [9], o Método de Feixes [35] e o Método do Volume [8]. Optamos pelo Método de Subgradientes, por ser o mais utilizado na literatura, dada a sua facilidade de implementação e eficiência.

Motivados pelos bons resultados relatados por Martinez e Cunha [40], que utilizaram com sucesso no problema MD-MST uma variação do método de subgradientes conhecida como *Deflected Subgradient Method* (DSM), proposta por Camerini et al. [13], resolvemos também implementar essa variação e comparar com o método clássico de subgradientes apresentado por Beasley [9].

No método clássico de subgradientes, a cada iteração k, dado o vetor de multiplicadores $u^k \geq 0$, encontramos uma solução x^k para $LP(u^k)$ e atualizamos os multiplicadores conforme

$$u_i^{k+1} = \max\{0, u_i^k + \theta^k s_i^k\},$$

onde s^k é o subgradiente, que é calculado da seguinte forma

$$s_i^k = d_i - \sum_{e \in \delta(i)} x_e^k,$$

a menos que esse valor seja negativo e $u_i^k = 0$, quando fazemos diretamente $s_i^k = 0$, e θ^k é o tamanho do passo calculado como

$$\theta^k = \pi(1{,}05UB - LP(u^k)) / \left\| s^k \right\|_2^2,$$

onde UB é o valor do melhor limite superior encontrado até o momento e π é um parâmetro que auxilia no controle to tamanho do passo. Usualmente, quando não se melhora o valor de $LP(u)$ após um certo número de iterações, o valor de π é reduzido.

No método clássico, a direção de atualização dos multiplicadores é dada pelo próprio subgradiente, ou seja, $u_i^k + \theta^k s_i^k$. Já no DSM, essa direção, que vamos denotar por β^k, depende não só do subgradiente, como também da direção anterior β^{k-1}, sendo dada pela expressão

$$\beta^k = (1 - \lambda)s^k + \lambda\beta^{k-1}$$

onde λ é um valor real no intervalo $(0, 1)$. Na primeira iteração, $\beta^0 = s^0$.

Os multiplicadores são atualizados como

$$u_i^{k+1} = \max\{0, u_i^k + \theta^k \beta_i^k\},$$

onde o tamanho do passo é calculado de uma forma diferente,

$$\theta^k = \pi(UB - LB) / \left\| \beta^k \right\|_2^2,$$

sendo UB e LB os valores do melhor limite superior e inferior encontrados até o momento.

Detalhes da nossa implementação do método de subgradientes clássico e DSM para o MDF-MSF podem ser vistos no algoritmo abaixo.

Algoritmo 5.3: *Método de Subgradientes para o MDF-MST*

Entrada: Parâmetros de Entrada: (π, T_π, N, u_0)

1. $\bar{x}^*$ //*Melhor solução primal*;
2. Enquanto condição de parada não for satisfeita, faça:
3. $\quad x^* \leftarrow$ *Resolve_Problema_Lagrangeano*(u^k);
4. $\quad$ Se $f(x^*) \geq z_{LB}$, faça:
5. $\quad\quad z_{LB} \leftarrow f(x^*)$;
6. $\quad\quad n_{stuck} \leftarrow 0$;
7. $\quad$ Caso contrário faça:
8. $\quad\quad n_{stuck} \leftarrow n_{stuck} + 1$;
9. $\quad \bar{x} \leftarrow$ *Heurística_Lagrangeana*(x^*); //*Solução primal viável.*
10. $\quad$ Se o valor de $f(\bar{x}) \leq z_{UB}$, faça:
11. $\quad\quad z_{UB} \leftarrow f(\bar{x})$;
12. $\quad\quad \bar{x}^* \leftarrow \bar{x}$;
13. $\quad$ Se $(z_{UB} - z_{LB} \leq 0.9)$ //*Encontramos a solução ótima dentro da margem.*
14. $\quad\quad$ PARE;
15. $\quad u^{k+1} \leftarrow$*Atualiza_Multiplicadores*$(\pi, z_{LB}, z_{UB}, u^k, x^*)$;
16. $\quad$ Se $n_{stuck} = N$, faça:
17. $\quad\quad n_{stuck} \leftarrow 0$;
18. $\quad\quad \pi \leftarrow \pi/2.0$;
19. $\quad\quad$ Se $\pi \leq T_\pi$:
20. $\quad\quad\quad$ PARE;
21. $\quad k \leftarrow k + 1$;
22. Retorne $\bar{x}^*$, z_{UB}, z_{LB};

5.3.3 Heurísticas Lagrangeanas

A cada iteração do método de subgradientes, temos uma solução do problema lagrangeano, que é uma árvore geradora de $G = (C \cup T, E)$ onde todo vértice em T é folha, ou seja, uma solução ótima de MST-F com os custos langrangeanos correntes. Para transformar essa árvore em uma solução viável do MDF-MST, podemos aplicar um Procedimento de Viabilidade.

À medida que o método de subgradientes atualiza os multiplicadores e, consequentemente, os custos lagrangeanos, a solução do problema lagrangeano tende a ter uma maior

quantidade de vértices com sua restrição de grau satisfeita. Com isso, nossa expectativa é que boas soluções comecem a aparecer na vizinhança da solução do problema lagrangeano e que, ao aplicarmos um Procedimento de Viabilidade, possamos obter limites superiores próximos da solução ótima ou mesmo a solução ótima.

Com esse pensamento em mente, desenvolvemos três heurísticas lagrangeanas. Na primeira heurística lagrangeana (Heurística H1), partimos da solução de MST-F do problema lagrangeano e aplicamos o Procedimento de Viabilidade PV1. As trocas de arestas são realizadas considerando os custos originais das arestas, ou seja, $c_e \ \forall e \in E$. A segunda heurística (Heurística H2) é similar à primeira, diferindo apenas no fato de que as trocas são efetuadas usando os custos lagrangeanos, isto é, $c_e - u_i - u_j \ \forall e = (i,j) \in E(C)$ e $c_e - u_i \ \forall e = (i,t) \in \delta(C : T)$. A Heurística H2 aproveita assim toda a informação fornececida pelo problema lagrangeano.

Em ambas, H1 e H2, logo após obtermos uma solução viável, com a aplicação do Procedimento de Viabilidade, aplicamos também a Busca Local Baseada em Trocas de Arestas e depois a busca local VND. Essas duas buscas locais pesquisam estruturas de vizinhanças distintas. Depois de alguns experimentos, resolvemos aplicar apenas a Busca Local Baseada em Trocas de Arestas incidentes nos terminais. Isso devido ao alto custo computacional que a Busca Local Baseada em Trocas de Arestas incidentes a centrais exige.

Também resolvemos aplicar, ao final do algoritmo, na melhor solução viável encontrada, o algoritmo de fluxo de custo mínimo que encontra a melhor distribuição de arestas entre os centrais e terminais, dada uma árvore em $G(C)$.

Os experimentos computacionais realizados atestam a boa qualidade dos limites superiores obtidos pelas heurísticas H1 e H2, como mostramos na Seção 5.2. Porém, a incorporação de qualquer delas ao método de subgradientes leva a um aumento significativo do tempo de computação, chegando à situação indesejável de que o algoritmo lagrangeano, nas maiores instâncias, pára por exceder o tempo computacional máximo estabelecido e não por atingir um dos critérios de convergência mostrados no Algoritmo 5.3. Executando poucas iterações do método de subgradientes, a consequência é a redução na qualidade dos limites inferiores gerados, o que comprometeria, por exemplo, o seu uso como ingrediente de um algoritmo Branch-and-Bound.

Verificamos que a parte dessas duas heurísticas que mais consome tempo computacional é o Procedimento de Viabilidade. Devido a isso, resolvemos desenvolver um outro procedimento, mais rápido, para encontrar soluções viáveis a partir das informações lagrangeanas. Mesmo que esse procedimento mais leve não encontre em uma aplicação isolada uma boa solução, a expectativa é que sua aplicação repetida, ao longo do processo lagrangeano, possa produzir boas soluções. Nossa ideia é apenas aplicar essa nova heurística lagrangeana nas instâncias onde H1 e H2 pararam por execeder o tempo máximo computacional. Esperamos que esse novo procedimento, ao permitir que o algoritmo de subgradientes execute mais iterações, possa até mesmo encontrar soluções melhores, em comparação às duas primeiras heurísticas lagrangeanas.

Com o desejo que essa nova heurística lagrangeana (Heurística H3) possa utilizar ao máximo a informação do problema lagrangeano e tenha baixo custo computacional, resolvemos

aplicar essecialmente Kruskal, considerando os custos lagrangeanos, em duas etapas.

Na primeira etapa, procuraremos encontrar uma árvore geradora em $G(C)$, de menor custo, que possa ser estendida para uma solução viável para MDF-MST com a conexão dos terminais aos centrais, da seguinte forma. Primeiro, ordenamos as arestas entre os centrais em ordem não decrescente dos custos lagrangeanos. Percorrendo essa ordem, uma aresta será adicionada à árvore se não gerar ciclo com as arestas já adicionadas e se, após sua adição, ainda for possível encontrar uma solução viável para o MDF-MST. Para verificar essa segunda condição, devemos determinar se a quantidade de ligações ainda disponíveis são suficientes para satisfazer as restrições de grau e gerar uma solução viável para o MDF-MST.

Para isso, seja H a floresta obtida até o momento. Seja $K(H) = \sum_{i \in C} \max\{0, d_i - d_H(i)\}$, onde $d_H(i)$ é o grau de i em H, o total de grau que ainda falta ser adicionado para cumprir as restrições de grau. Seja $M(H) = 2(c-1) - \sum_{i \in C} d_H(i) + t$ o grau remanescente que ainda pode ser aportado às restrições de grau mínimo com o número de ligações que ainda devem ocorrer para formar a árvore. Note que $2(c-1)$ e t são, respectivamente, o aporte final para tais restrições fornecido pelas ligações entre centrais e entre centrais e terminais. Dada uma aresta $(i, j) \in E(C)$, seja k_{ij} a quantidade de centrais em $\{i, j\}$ cuja restrição de grau está violada, antes de adicionarmos a aresta (i, j). Então, $k_{ij} \in \{0, 1, 2\}$.

A segunda condição para que uma aresta seja adicionada em H será então $M(H) - 2 \geq K - k_{ij}$. Se essa condição for verdadeira, então a aresta (i, j) será adicionada à solução, gerando uma nova floresta H'. Note que $K(H') = K(H) - k_{ij}$ e $M(H') = M(H) - 2$. Com isso, garantimos que a segunda etapa da heurística encontre uma solução viável, considerando que existem todas as arestas entre C e T (como em nossas instâncias).

Na segunda etapa, ordenamos as arestas entre os centrais e terminais considerando os custos lagrangeanos em ordem não decrescente e percorremos as arestas nesta ordem. Uma aresta será adicionada à árvore se o terminal em que essa aresta incide ainda não foi conectado a um central.

O custo computacional dessa nova heurística lagrangeana é praticamente, no pior caso, percorrer toda a lista de arestas, já que para resolver o problema lagrangeano já ordenamos as arestas pelo custo lagrangeano.

5.3.4 Resultados Computacionais

Aqui iremos apresentar com mais detallhes os resultados obtidos pela relaxação lagrangeana para o MDF-MST.

Para cada heurística lagrangeana, implementamos um algoritmo lagrangeano que utiliza o método de subgradientes para encontrar limites inferiores. Iremos chamar de Lagrangeano 1, Lagrangeano 2 e Lagrangeano 3 os algoritmos lagrangeanos que utilizam a heurística H1, H2 e H3, respectivamente.

Para os multiplicadores lagrangeanos iniciais, fizemos $u_i = 1/|C|$, $\forall i \in C$. Para os outros parâmetros do algoritmo usamos: $\pi = 2$, $T_\pi = 0.1$, $N = 50$ e $\lambda = 0.05$.

Algoritmos	Média GAP	Desvio Padrão GAP	Média Tempo (s)
Lagrangeano 1	0,071273888	0,160514743	4139,48
Lagrangeano 2	0,055735131	0,130075902	4043,93
Lagrangeano 3	0,021375353	0,016444121	1027,18

Tabela 5.8: Média e desvio padrão dos GAP e média dos tempos para os algoritmos lagrangeano para as todas instâncias da classe ALM

Algoritmos	Média GAP	Desvio Padrão GAP	Média Tempo (s)
Lagrangeano 1	0,070375705	0,117114791	4190,26
Lagrangeano 2	0,068289631	0,119923941	4195,25
Lagrangeano 3	0,029047311	0,021433223	1744,95

Tabela 5.9: Média e desvio padrão dos GAP e média dos tempos para os algoritmos lagrangeano para as todas instâncias da classe NEU

Nas Tabelas 5.8 e 5.9 estão a média e o desvio padrão dos GAP's para todas as instâncias. O GAP foi calculado como sendo $(UB - LB)/LB$, onde UB é a melhor solução viável encontrada pela heurística lagrangeana e LB o melhor limite inferior encontrado pelo método de subgradientes.

Ao analisarmos essas duas tabelas, verificamos que o Lagrangeano 3 foi em média melhor que os outros dois algoritmos lagrangeano. Podemos explicar esse resultado pelo fato do Lagrangeano 1 e 2, para as instâncias a partir de 700 centrais, pára por exceder o limite máximo de tempo permitido e em muitos casos com poucas iterações executadas. Logo, o limite inferior e superior obtidos por esses algoritmos não são bons, principalmente o limite inferior, como podemos verificar nas tabelas 5.16 e 5.17. Comparando as médias dos tempos, verificamos que o Lagrangeano 1 e 2 são equivalentes e o Lagrangeano 3 é notadamente mas rápido.

Resolvemos realizar a mesma comparação anterior, dessa vez apenas analisando as instâncias em que todos os algoritmos param sem exceder o limite de tempo máximo. Nas tabelas 5.10 e 5.11 estão a média e o desvio padrão dos GAP's para essas instâncias. Observamos que para as instâncias ALM, o Lagrangeano 2 possui a melhor média dos Gaps, e para as instâncias NEU o Lagrangeano 3 se mostra melhor.

Agora, iremos realizar uma comparação dos valores dos limites superiores obtidos pelas heurísticas lagrangeanas. Iremos considerar apenas as instâncias que sabemos o valor de sua solução ótima. O Gap será calculado como $(UB - w)/w$, onde UB é o limite superior e w a solução ótima. Na tabelas 5.12 e 5.13 estão a média e o desvio padrão ddos Gap's para essas instâncias. A heurística H2 possui o melhor Gap para as instâncias ALM e H3 para as instâncias NEU.

Algoritmos	Média GAP	Desvio Padrão GAP	Média Tempo (s)
Lagrangeano 1	0,012013394	0,016604108	1612,81
Lagrangeano 2	0,009605209	0,013070932	1491,35
Lagrangeano 3	0,015262505	0,010162806	223,83

Tabela 5.10: Média e desvio padrão dos GAP e média dos tempos para os algoritmos lagrangeano para as algumas instâncias da classe ALM

Algoritmos	Média GAP	Desvio Padrão GAP	Média Tempo (s)
Lagrangeano 1	0,032146621	0,047580494	1866,63
Lagrangeano 2	0,031073308	0,047729227	1824,03
Lagrangeano 3	0,024913056	0,019013422	490,29

Tabela 5.11: Média e desvio padrão dos GAP e média dos tempos para os algoritmos lagrangeano para as algumas instâncias da classe NEU

Heurísticas	Média GAP	Desvio Padrão GAP
H1	0,0073288505	0,011993317
H2	0,0056496776	0,009893136
H3	0,0091002857	0,008049812

Tabela 5.12: Média e desvio padrão dos GAP para as heurísticas lagrangeanas para as instâncias da classe ALM com ótimo conhecido

Nas Tabelas 5.16 e 5.17 temos os valores dos limites inferiores e superiores obtidos por cada um dos algoritmos para cada uma das instâncias. Para cada algoritmo lagrangeano, temos a coluna UB, LB e T(s) que representam o limite superior, limite inferior e o tempo em segundos, respectivamente. Para cada instância, o melhor limite superior encontrado está em negrito.

Fazendo uma comparação quantitativa com relação à qualidade dos limites superiores, verificamos que das 40 instâncias ALM, o Lagrangeano 1 foi melhor em 18, o Lagrangeano 2 em 20, e o Lagrangeano 3 em 16. Quando analisamos as 40 instâncias NEU, o Lagrangeano 1 foi melhor em 18, o Lagrangeano 2 em 16 e o Lagrangeano 3 em 17. Limites superiores iguais foram computados nessa comparação para ambos os algoritmos.

Fazendo agora uma análise do Lagrangeano 3 apenas para as instâncias que os dois outros algoritmos pararam por exceder o limite de tempo, verificamos que ele foi melhor em 8 das 14 instâncias ALM e em 6 das 12 instâncias NEU em que foi executado. Apesar do Lagrangeano 3 ser melhor em termos quantitativos, ele ainda ficou distante do que esperávamos, já que ele executa uma quantidade muito maior de iterações de subgradientes, comparada com os outros algoritmos lagrangeanos, e em nenhum momento ele pára por exceder o limite máximo de tempo.

Os tempos computacionais do Lagrangeano 1 e 2 são excessivos, principalmente para instâncias a partir 600 vértices centrais. O Lagrangeano 1 pára em 25 instâncias por exceder o limite máximo de tempo e o Lagrangeano 2 pára em 24.

Realizamos uma comparação dos resultados obtidos dos algoritmos lagrangeano 2 e 3 para com o algoritmo planos de corte com o cplex da Seção 5.1. Nas Tabelas 5.14 e 5.15 estão

Heurísticas	Média GAP	Desvio Padrão GAP
H1	0,0159191	0,0283782481
H2	0,0151991	0,0229481723
H3	0,0140798	0,0121691858

Tabela 5.13: Média e desvio padrão dos GAP para as heurísticas lagrangeanas para as instâncias da classe NEU com ótimo conhecido

Instâncias	Planos de Corte - Cplex	Lagrangeano 2		Lagrangeano 3	
#*c*	Média Tempo(s)	Média Tempo(s)	Média Gap	Média Tempo(s)	Média Gap
60 e 100	5,31	5,84	0,000535299	3,10	0,003837469
200	398,14	120,43	0,013097594	31,82	0,02435873
300	1610,36	321,91	0,017875805	78,28	0,019612095
400	*	1294,98	0,011250682	210,63	0,018838895
500	*	2564,70	0,013213583	432,35	0,017101014
600	*	7856,58	0,015037507	832,28	0,020337558
700	*	9094,76	0,06766017	1637,21	0,030979877
800	*	9362,50	0,163534449	2536,69	0,036710401
900	*	9811,77	0,254610919	4506,44	0,038140022

Tabela 5.14: Comparação entre os algoritmos lagrangeano e o planos de corte para as instâncias da classe ALM

Instâncias	Planos de Corte - Cplex	Lagrangeano 2		Lagrangeano 3	
#*c*	Média Tempo(s)	Média Tempo(s)	Média Gap	Média Tempo(s)	Média Gap
60 e 100	5,43	5,33	0,004967725	4,84	0,012372388
200	392,87	97,07	0,026410715	50,00	0,022666332
300	4072,25	564,33	0,029920761	171,24	0,025231588
400	*	1669,75	0,035375419	498,49	0,029479726
500	*	3718,80	0,056947323	943,40	0,034763586
600	*	6707,60	0,058923491	1759,26	0,037505381
700	*	9287,13	0,103302402	2884,75	0,037063648
800	*	9610,86	0,144033998	4240,06	0,038628685
900	*	10286,28	0,218046753	6892,68	0,040389385

Tabela 5.15: Comparação entre os algoritmos lagrangeano e o planos de corte para as instâncias da classe NEU

os resultados dessa comparação. Para o algoritmo de planos de corte, exibimos a média dos tempos computacionais para as instâncias que ele encontrou a solução ótima. O asterisco significa que ele nao encontrou a solução ótima e nem mesmo relaxação linear. Para o lagrangeano 2 e 3, exibimos a média dos tempos e Gap's, obtidos por esses algoritmos. Observamos que para as instâncias que o planos de corte não encontra nem mesmo a relaxação linear, os lagrangeanos encontram limites superiores e inferiores de boa qualidade. Acreditamos que através de outros procedimentos que encontre solução de boa qualidade a partir do problema lagrangeano, assim como fizemos com o lagrangeano 3, possamos melhorar ainda mais os resultados dos limites superiores.

Instâncias	Lagrangeano 1			Lagrangeano 2			Lagrangeano 3		
	LB	UB	T(s)	LB	UB	T(s)	LB	UB	T(s)
ALM60-1	6942,73	**6943**	0.24	6942,37	**6943**	0.24	6942,70	**6943**	1,57
ALM60-2	6771,99	**6772**	0.26	6771,59	**6772**	0.23	6769,35	6791	1,12
ALM60-3	6708,19	**6709**	0.81	6708,35	**6709**	0.49	6708,22	**6709**	0,37
ALM60-4	7039,43	**7040**	1.05	7039,36	**7040**	0.77	7039,89	**7040**	0,44
ALM100-1	8209,38	8219	5.65	8209,35	**8217**	6.00	8167,82	8247	3,77
ALM100-2	8065,40	**8069**	8.61	8065,39	**8069**	9.96	8053,03	8159	3,75
ALM100-3	8053,37	**8055**	14.15	8053,00	**8055**	13.75	8052,58	8071	5,89
ALM100-4	8123,83	**8139**	15.89	8120,20	**8139**	15.29	8130,22	8148	7,89
ALM200-1	11508,15	**11509**	35.53	11508,24	**11509**	35.89	11483,50	11617	30,64
ALM200-2	11286,83	**11317**	138.00	11286,64	**11317**	100.05	11238,35	11539	25,6
ALM200-3	11302,13	**11654**	125.01	11321,53	11565	151.53	11317,59	11569	25,11
ALM200-4	11194,62	11592	164.70	11221,35	**11537**	194.23	11175,26	11587	45,93
ALM300-1	13456,82	13465	465.36	13456,80	**13464**	334.84	13412,21	13594	116,82
ALM300-2	13002,44	**13004**	280.24	13002,38	**13004**	258.46	12972,26	13090	109,35
ALM300-3	12736,36	13135	466.70	12731,16	13154	425.86	12732,19	**13019**	48,91
ALM300-4	12445,34	12910	343.97	12436,99	12905	268.48	12438,89	**12853**	38,04
ALM400-1	15513,72	**15522**	1081.00	15513,52	15526	955.18	15421,22	15686	317,01
ALM400-2	15106,72	**15117**	882.71	15106,47	**15117**	827.98	15039,58	15234	274,6
ALM400-3	15282,32	16001	1378.71	15305,39	15659	1658.91	15319,94	**15639**	150,88
ALM400-4	15177,12	15750	1757.01	15185,26	**15495**	1737.85	15164,50	15535	100,02
ALM500-1	17290,40	17338	3149.48	17290,84	**17328**	2054.62	17236,34	17430	727,62
ALM500-2	17153,05	**17161**	2360.85	17152,61	17162	2493.12	17077,14	17353	612,57
ALM500-3	16946,05	17390	2534.42	16938,35	17360	2109.52	16951,48	**17231**	172,48
ALM500-4	17625,63	18095	4302.26	17617,89	18063	3601.52	17615,97	**18048**	216,73
ALM600-1	18964,25	**18993**	7944.87	18963,73	19012	7735.19	18863,10	19159	1327,55
ALM600-2	18645,96	18658	6985.16	18645,72	**18652**	7045.84	18548,78	18889	1310,12
ALM600-3	19255,92	19978	9103.45	19278,45	19865	8230.73	19275,74	**19810**	368.76
ALM600-4	19174,47	19825	8587.89	19184,05	19699	8414.54	19199,59	**19576**	322.67
ALM700-1	20373,08	**20403**	9093.16	20374,77	20417	9049.19	20283,35	20577	2958.51
ALM700-2	19892,31	**19938**	9062.37	19894,49	19949	8986.43	19816,25	20191	2502.65
ALM700-3	19569,76	22591	9230.17	20072,18	21894	9220.05	20660,41	**21364**	621.85
ALM700-4	18460,19	23199	9236.79	18851,74	22152	9123.36	20065,76	**21199**	465.83
ALM800-1	21493,60	21678	9230.96	21572,63	**21618**	9180.35	21489,05	21809	3937.66
ALM800-2	21279,31	21351	9219.43	21270,26	**21345**	9227.30	21187,21	21736	4126.36
ALM800-3	20417,31	26299	9684.02	21024,40	25365	9503.54	23171,89	**23925**	1194.97
ALM800-4	19091,56	30114	9487.33	19933,21	28745	9538.82	24514,94	**26318**	887.75
ALM900-1	22649,44	22884	9578.01	22648,10	**22856**	9620.99	22593,75	23036	8600.06
ALM900-2	21953,99	**22312**	9691.07	22055,67	22276	9457.03	22054,33	22504	6288.82
ALM900-3	19912,83	31310	10093.33	20264,94	30285	9997.07	25924,31	**27254**	1730.88
ALM900-4	18954,32	30354	9838.94	19633,56	29545	10172.00	24912,72	**26440**	1405.98

Tabela 5.16: Resultado dos algoritmos lagrangeano para as instâncias da classe ALM

Instâncias	Lagrangeano 1			Lagrangeano 2			Lagrangeano 3		
	LB	UB	T(s)	LB	UB	T(s)	LB	UB	T(s)
NEU60-1	3031,35	**3032**	0.48	3031,45	**3032**	0.68	3030,82	3054	2,34
NEU60-2	3872,00	**3872**	0.41	3871,81	**3872**	0.91	3871,98	**3872**	2,85
NEU60-3	2969,77	2983	1.95	2969,62	**2973**	2.55	2966,97	3032	1,19
NEU60-4	4030,38	**4031**	1.08	4030,13	**4031**	1.10	4029,88	4071	1,86
NEU100-1	3078,33	**3079**	5.06	3078,22	**3079**	5.32	3071,05	3158	7,2
NEU100-2	3857,10	**3858**	5.96	3857,95	3859	10.60	3852,31	**3858**	12,98
NEU100-3	3078,55	3164	14.31	3079,41	3152	7.74	3076,48	**3107**	3,13
NEU100-4	3648,01	**3695**	14.06	3647,72	3699	13.75	3636,13	3707	7,19
NEU200-1	3325,15	**3326**	28.39	3325,31	**3326**	39.93	3311,48	3336	45,37
NEU200-2	4279,82	**4284**	149.11	4279,28	**4284**	116.04	4257,64	4311	102,95
NEU200-3	3376,38	**3482**	99.88	3374,47	3579	74.30	3369,51	3501	19,45
NEU200-4	4213,10	**4315**	150.28	4213,77	4398	158.02	4195,00	4328	32,22
NEU300-1	3461,32	**3462**	317.22	3461,84	3464	476.90	3441,79	3476	178,94
NEU300-2	4381,86	**4382**	565.97	4381,90	**4382**	648.94	4361,44	4406	295,29
NEU300-3	3309,08	3660	472.11	3305,51	3532	538.12	3290,94	**3456**	63,7
NEU300-4	3985,65	4254	479.16	3988,51	4190	593.37	3973,35	**4095**	147,04
NEU400-1	3546,66	3550	1149.25	3546,65	**3549**	1330.00	3514,79	3578	551,23
NEU400-2	4219,76	4223	2022.44	4219,48	**4222**	2290.31	4185,47	4238	953,46
NEU400-3	3467,80	3863	1430.09	3470,41	3826	1320.19	3452,00	**3655**	119,33
NEU400-4	4174,96	4377	1397.12	4180,07	4338	1738.50	4161,08	**4280**	369,93
NEU500-1	3509,77	**3511**	3027.79	3509,34	3513	4188.37	3463,00	3539	1092,64
NEU500-2	4359,01	**4366**	4109.62	4358,68	**4366**	3405.68	4324,87	4390	1704,75
NEU500-3	3422,05	3829	4433.69	3420,32	3930	3998.53	3395,37	**3644**	325,33
NEU500-4	4301,22	4668	3662.88	4304,62	4632	3282.61	4286,46	**4410**	650,89
NEU600-1	3662,30	**3670**	5549.29	3661,56	3676	5851.40	3611,22	3724	1674,72
NEU600-2	4542,90	4556	8350.73	4543,01	**4554**	6899.62	4502,43	4591	4015,32
NEU600-3	3443,36	4027	7101.95	3442,07	4064	4854.18	3416,73	**3658**	394,59
NEU600-4	4408,86	4758	7725.45	4405,69	4620	9225.20	4384,02	**4509**	952,41
NEU700-1	3798,28	3826	9144.57	3801,00	**3823**	9180.44	3763,83	3866	3038.43
NEU700-2	4749,11	**4765**	9135.63	4751,67	4766	9337.32	4715,11	4785	6137.39
NEU700-3	3517,77	4414	9180.71	3513,69	4507	9147.93	3621,14	**3885**	788.60
NEU700-4	4290,47	5142	9620.04	4459,26	5002	9482.83	4519,94	**4671**	1574.57
NEU800-1	3918,83	3971	9140.29	3931,23	**3962**	9205.33	3890,43	3978	4785.05
NEU800-2	4911,09	**4950**	9662.49	4900,55	**4950**	10142.78	4880,21	4961	8360.44
NEU800-3	3503,23	4619	9330.99	3513,97	4466	9326.72	3652,80	**3956**	1177.54
NEU800-4	4336,03	5412	9916.04	4352,56	5603	9768.61	4662,70	**4814**	2637.20
NEU900-1	4079,15	**4184**	9583.92	4068,95	4186	9843.66	4092,17	4195	7811.60
NEU900-2	4960,51	5074	10039.44	4970,98	**5070**	11264.88	4969,97	5088	14615.98
NEU900-3	3243,00	5036	9670.55	3243,00	5175	9490.93	3879,23	**4179**	1733.22
NEU900-4	4390,13	5539	10920.12	4435,73	5446	10545.63	4806,80	**4977**	3409.92

Tabela 5.17: Resultado dos algoritmos lagrangeano para as instâncias da classe NEU

5.4 Decomposição de Benders

Normalmente, a quantidade de memória e tempo computacional necessários para resolver um problema de otimização combinatória aumenta significativamente com o aumento do número de variáveis e restrições. Métodos de decomposição têm sido usados para tratar problemas desse tipo. Dentre esses métodos, destacamos a Decomposição de Benders [10] como uma estratégia com bons resultados reportados na literatura.

De uma forma geral, a decomposição de benders divide o problema em dois problemas de dimensões menores. O primeiro problema, chamado de Problerma Mestre, determina o valor de um subconjunto de variáveis; o restante das variáveis será determinado pelo problema resultante do problema original quando fixamos essas variáveis definidas pelo problema mestre. O segundo problema é chamado de Subproblema Escravo. Normalmente, a decomposição de benders é aplicada em problemas de programação inteira mista, onde o problema mestre vai lidar com as variáveis inteiras e o subproblema escravo, com as variáveis contínuas. Embora menos frequente na literatura, também é possível sua aplicação a problemas de programação inteira pura.

A seguir apresentamos brevemente essa decomposição. Considere o seguinte problema de programação mista:

$$\begin{aligned} (P) \qquad \min \quad & d^T x + c^T y \\ s.a: \quad & Ax + By \geq b \\ & x \geq 0, y \in Y \subseteq \mathbb{Z}^q \end{aligned}$$

onde x e y são variáveis contínuas e inteiras, respectivamente, Y é o conjunto de pontos inteiros de um poliedro, A, B são matrizes e b, c e d são vetores. Os vetores e as matrizes possuem dimensões apropriadas. Suponha que as variáveis y são as "variáveis complicantes", ou seja, quando fixamos um valor para y o problema resultante torna-se fácil. Por simplicidade, vamos supor que P tenha solução ótima.

Como falamos anteriormente, a decomposição de benders irá criar dois problemas distintos, a partir do problema P. O problema mestre irá conter as variáveis y e o subproblema irá conter as variáveis x.

Seja $y \in Y$. Então, definindo

$$\begin{aligned} P(y) \qquad g(y) = \min \quad & d^T x \\ s.a.: \quad & Ax \geq b - By \\ & x \geq 0 \end{aligned}$$

o problema original pode ser rescrito como

$$\begin{aligned} P \qquad \min \quad & c^T y + g(y) \\ s.a.: \quad & y \in Y \subseteq \mathbb{Z}^q \end{aligned}$$

O problema $P(y)$ é um problema linear. Considere o seu dual.

$$\begin{array}{lll} D(y) & \max & \beta^T(b-By) \\ & s.a.: & A^T\beta \leq d \\ & & \beta \geq 0 \end{array}$$

Note que, para qualquer $y \in Y$, $P(y)$ não pode ser ilimitado; caso contrário P seria ilimitado. Já $P(y)$ iniviável para todo $y \in Y$ implicaria P inviável. Portanto, $P(y)$ deve possuir solução ótima para algum $y \in Y$. Pelo Teorema Fundamental da Dualidade, $D(y)$ tem solução ótima para algum $y \in Y$ e, portanto, é viável para todo y. Assim, temos que $P(y)$ não é inviável se, e somente se, $D(y)$ não é ilimitado.

Seja $\Delta = \{\beta \geq 0 : A^T\beta \leq d\}$ o conjunto viável de $D(y)$ (que independe de y). Sejam $(\alpha^1, ..., \alpha^p)$ e $(\theta^1, ..., \theta^r)$ seus pontos extremos e raios extremos, respectivamente. Note que $P(y)$ não é inviável se, e somente se,

$$\theta_j^T(b-By) \leq 0, \forall j = 1, .., r$$

E se $P(y)$ tem solução ótima,

$$g(y) = \max\{\alpha_i^T(b-By) : i = 1, 2, \ldots, p\}.$$

Logo, podemos reescrever P como:

$$\begin{array}{lll} P & \min & c^T y + g \\ & s.a.: & g \geq \alpha_i^T(b-By), \forall i = 1, ..., p \quad (5.1) \\ & & \theta_j^T(b-By) \leq 0, \forall j = 1, .., r \quad (5.2) \\ & & y \in Y \text{ e } f \text{ livre} \end{array}$$

Essa reformulação do problema P é o problema mestre de benders, e o problema dual $D(y)$ é o subproblema de benders.

O algoritmo de benders itera resolvendo problemas mestre de benders e depois subproblema de benders, até que uma condição de parada seja atingida. Na iteração inicial, o problema mestre pode ter um subconjunto finito de restrições (5.1) e/ou (5.2), ou mesmo obter o valor de y inicial por meio de uma heurística. A cada iteração, teremos a solução ótima do problema mestre $(\bar{y}, \bar{g})$ e a resolução ótima do subproblema relacionado $D(\bar{y})$. Se o Subproblema $D(\bar{y})$ for ilimitado, então uma restrição do tipo (5.2) é adicionada ao problema mestre. Caso contrário, $D(\bar{y})$ tem solução ótima $\bar{\beta} \in \{\alpha_1, \ldots, \alpha_p\}$. Se $\bar{g} \geq \bar{\beta}^T(b-B\bar{y})$, então a solução ótima foi encontrada; senão, uma restrição do tipo (5.1) é adicionada ao problema mestre.

As restrições do tipo (5.2) são chamadas de Cortes de Viabilidade de Benders, porque elas fornecem condições para a viabilidade do problema P. Já as restrições do tipo (5.1) são chamadas de Cortes de Otimalidade de Benders, porque elas fornecem condições para a otimalidade do problema P.

A cada iteração, temos um limite superior e um limite inferior para P dados por $c^T\overline{y} + g(\overline{y})$ e $c^T\overline{y} + \overline{g}$, respectivamente. Quando, numa iteração, encontramos $g(\overline{y}) = \overline{g}$, então encontramos a solução ótima do problema original. Outros dois critérios de parada que podem ser utilizados são a quantidade máxima de iterações e o tempo computacional máximo. Sabemos que a cada iteração o valor do limite inferior é sempre maior ou igual ao da iteração anterior, mas não existe essa garantia para o limite superior.

Como as quantidades p e r são finitas e a cada iteração um corte é adicionado ao problema mestre, então o método converge para a solução ótima em uma quantidade de iterações finita. Para maiores detalhes e a demostração da convergência do método, recomedamos uma leitura a Benders [10].

5.4.1 Decomposição de Benders para o MDF-MST

Como provado na Proposição 3.11, quando o grafo $G(C)$ é uma árvore, encontrar a solução ótima do MDF-MST é polinomial e se resume a conectar os centrais com os terminais de forma a satisfazer as restrições de grau a custo mínimo. Dessa forma, uma maneira diferente de visualizar o MDF-MST é encontrar uma árvore geradora entre os centrais de forma que, ao conectar os terminais, tenhamos uma árvore de custo mínimo que satisfaça as restrições de grau. Como visto na seção sobre a Complexidade Parametrizada, poderíamos gerar árvores geradoras entre centrais e depois conectar os terminais para tentar encontrar a melhor solução.

A formulação $\text{MDF}^r_{\text{cutsetD}}$ pode ser reformulada com base nesta concepção, ou seja, teremos dois problemas: o primeiro seria gerar uma árvore geradora entre os centrais; o segundo, dada essa árvore, conectar os centrais com os terminais de forma a satisfazer as restrições de grau. O objetivo no final é encontrar uma árvore geradora de G de custo mínimo satisfazendo as restrições de grau.

Seguindo essa ideia, vamos reescrever a formulação $\text{MDF}^r_{\text{cutsetD}}$, onde usaremos variáveis de decisão diferentes para indicar as arestas entre centrais e entre centrais e terminais. A variável y_{ij} será definida apenas para os arcos entre centrais (arcos em $A(C)$) e continuará indicando se o arco (i, j) estará ou não na solução. Já a variável z_{it}, definida para os arcos entre centrais e terminais, irá indicar quais desses arcos estarão na solução. Vamos denominar os custos dos centrais para os terminais por h_{it} e reservar c_{ij} apenas para os custos entre centrais.

Assim obtemos uma reformulação para o mdf-MST:

$$\text{(MDF-REFOR)} \quad \min \sum_{(i,j)\in A(C)} c_{ij}y_{ij} + \sum_{(i,j)\in\delta(C:T)} h_{ij}z_{ij}$$

$$\text{sujeito a:} \sum_{(i,j)\in A(C)} y_{ij} = |C| - 1 \tag{5.3}$$

$$\sum_{(i,j)\in\delta^+(S:C\setminus S)} y_{ij} \geq 1, \forall S \subseteq C \text{ e } r \in S \tag{5.4}$$

$$\sum_{(i,j)\in\delta^-(j)\cap A(C)} y_{ij} = 1, \forall j \in C\setminus\{r\} \tag{5.5}$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in A(C) \tag{5.6}$$

$$\sum_{(i,j)\in\delta^+(i)\cap A(C)} y_{ij} + \sum_{(i,j)\in\delta^+(i)\cap\delta(C:T)} z_{ij} \geq d_i - 1, \forall i \in C\setminus\{r\} \tag{5.7}$$

$$\sum_{(r,j)\in\delta^+(r)\cap A(C)} y_{rj} + \sum_{(r,j)\in\delta^+(r)\cap\delta(C:T)} z_{rj} \geq d_r \tag{5.8}$$

$$\sum_{(i,t)\in\delta^-(t)\cap\delta(C:T)} z_{it} = 1, \forall t \in T \tag{5.9}$$

$$z_{ij} \in \{0,1\}, \forall (i,j) \in \delta(C:T) \tag{5.10}$$

Seja $Y = \{y : y \text{ satisfaz } (5.3) - (5.6)\}$, ou seja, o conjunto que descreve todas as árvores geradoras de $G(C)$. Fixados valores para as variáveis y, as restrições (5.7) e (5.8) podem ser restruturadas da seguinte forma:

$$\sum_{(i,j)\in\delta^+(i)\cap\delta(C:T)} z_{ij} \geq d_i - 1 - \sum_{(i,j)\in\delta^+(i)\cap A(C)} y_{ij}, \forall i \in C\setminus\{r\} \tag{5.11}$$

$$\sum_{(r,j)\in\delta^+(r)\cap\delta(C:T)} z_{rj} \geq d_r - \sum_{(r,j)\in\delta^+(r)\cap A(C)} y_{rj} \tag{5.12}$$

Definimos o poliedro $Z(y) = \{z : z \text{ satisfaz } (5.11), (5.12), (5.9) \text{ e } (5.10)\}$. Assim, a reformulação pode ser vista como:

$$P \quad \min \; c^T y + h^T z \tag{5.13}$$

$$s.a. \; y \in Y \tag{5.14}$$

$$z \in Z(y) \tag{5.15}$$

Dado $y \in Y$ (descrevendo uma árvore geradora de $G(C)$), definimos o problema

$$P(y) \quad \min \left\{ h^T z : z \in Z(y) \right\}$$

e seja $g(y)$ o seu valor ótimo. Seja H_y a árvore correspondente a y e $d_y(i)$ o grau de i em H_y. Temos que

$$d_y(i) = \begin{cases} \sum_{(i,j)\in\delta^+(i)\cap A(C)} y_{ij} + 1, & i \in C \setminus \{r\}, \\ \sum_{(i,j)\in\delta^+(i)\cap A(C)} y_{ij}, & i = r \end{cases}$$

Definimos também

$$\Delta_y(i) = d_i - d_y(i), \forall i \in C$$

Assim, $Z(y)$ é definido por $z_{ij} \in \{0,1\}$ e por

$$\sum_{(i,j)\in\delta^+(i)\cap\delta(C:T)} z_{ij} \geq \Delta_y(i), \forall i \in C, \quad \text{e} \quad \sum_{(i,t)\in\delta^-(t)\cap\delta(C:T)} -z_{it} = -1, \forall t \in T. \tag{5.16}$$

A matriz de restrições (5.16) é a matriz de incidência vértice-arco do grafo (bipartido) induzido pelas arestas entre centrais e terminais. Então é Totalmente Unimodular. Logo, podemos substituir $z_{ij} \in \{0,1\}$ por $z_{ij} \geq 0$ na definição de $Z(y)$. Note que $z_{ij} \leq 1$ é garantido pelas igualdades em (5.16). Por conseguinte, $P(y)$ é um problema de programação linear, cujo dual é

$$D(y) \quad \max \sum_{i\in C} \Delta_y(i)u_i - \sum_{t\in T} v_t \tag{5.17}$$

$$\text{s. a.: } u_i - v_t \leq h_{it}, \forall (i,t) \in \delta(C:T) \tag{5.18}$$

$$u_i \geq 0, \forall i \in C,\ v_t \text{ livre}, \forall t \in T \tag{5.19}$$

Com a reformulação MDF–REFOR, podemos desenvolver um esquema de decomposição de Benders para o MDF-MST. O subproblema da decomposição de Benders é definido pelo dual $D(y)$. Seja Q e K os conjuntos de vértices e direções extremas de (5.18)–(5.19). O problema mestre é então definido como:

$$PM \quad \min \ c^T y + S$$

$$\text{s. a.: } S \geq \sum_{i\in C} \Delta_y(i)u_i - \sum_{t\in T} v_t, \forall (u,v) \in Q \tag{5.20}$$

$$\sum_{i\in C} \Delta_y(i)u_i - \sum_{t\in T} v_t, \forall (u,v) \in K \tag{5.21}$$

$$y \in Y \tag{5.22}$$

Substituindo a expressão de $\Delta_y(i)$, temos

$$\begin{aligned}\sum_{i\in C} \Delta_y(i)u_i &= \sum_{i\in C\setminus r} (d_i - 1)u_i + d_r u_r - \sum_{i\in C} \sum_{(i,j)\in\delta^+(i)\cap A(C)} y_{ij}u_i \\ &= \sum_{i\in C} (d_i - 1)u_i + u_r - \sum_{(i,j)\in A(C)} y_{ij}u_i\end{aligned}$$

Com isso, o problema mestre pode ser rescrito:

$$
\begin{aligned}
PM \quad \min \quad & c^T y + S \\
\text{s.a: } & S + \sum_{(i,j)\in A(C)} y_{ij} u_i \geq \sum_{i\in C} (d_i - 1) u_i + u_r - \sum_{t\in T} v_t, \forall (u,v) \in Q && (5.23) \\
& \sum_{(i,j)\in A(C)} y_{ij} u_i \geq \sum_{i\in C} (d_i - 1) u_i + u_r - \sum_{t\in T} v_t, \forall (u,v) \in K && (5.24) \\
& y \in Y && (5.25)
\end{aligned}
$$

Nosso algoritmo inicia com uma solução inicial $y^k \in Y$ e $S^k = -\infty$, para $k = 0$. Resolvemos o subproblema $D(y^k)$. Se o subproblema for ilimitado, encontramos uma direção extrema (u^k, v^k). Adicionamos o corte de viabilidade por ela definido ao problema mestre *PM*. Caso a solução seja limitada, encontramos um vértice ótimo (u^k, v^k). Se o teste de otimalidade não for satisfeito, acrescentamos o corte de otimalidade definido por esse ponto extremo ao problema mestre *PM*.

Em qualquer um dos casos, após adicionarmos um corte, resolvemos novamente o problema mestre, obtendo a próxima solução (S^{k+1}, y^{k+1}), e repitimos o processo acima descrito. Exceto pela primeira iteração, ou seja, $k = 0$, o critério de parada é o valor do subproblema $D(y^k)$ ser igual a S^k. Também adicionamos como critério de parada o tempo computacional máximo.

Para resolver o subproblema e o problema mestre de benders estamos usando o solver CPLEX. Resolver o problema mestre consiste em encontrar uma árvore geradora de $G(C)$ que satisfaça os cortes de otimalidade e de viabilidade. Depois de alguns experimentos computacionais e devido às experiências com o algoritmo de planos de corte e com o algoritmo exato usando o CPLEX, optamos por iniciar a resolução do mestre resolvendo primeiro sua relaxação linear, ou seja, com as variáveis y relaxadas. Uma vez incluído um corte de benders no mestre, utilizamos o algoritmo de planos de corte para gerar as DCUTs violadas, com as mesmas políticas de inserção e remoção desses cortes já vistas. Ressaltamos que as restrições de corte (5.4) presentes no problema mestre corrente, quando do final de sua solução, são levadas para o problema mestre seguinte. Da mesma forma que no algoritmo exato com o CPLEX, quando encontramos a solução ótima da relaxação linear, convertemos as variáveis y para inteiros e deixamos de remover restrições de corte (5.4) acrescidas durante as próximas iterações.

5.4.2 Resultados Computacionais

Os resultados da decomposição de benders para o MDF-MST estão na Tabela 5.18. Estamos fazendo uma comparação entre o tempo obtido pelo benders e o tempo do algoritmo da Seção 5.1. Na coluna "Cplex T(s)" temos o tempo do algoritmo da Seção 5.1 e na coluna "Benders T(s)" temos o tempo do algoritmo de Benders para o MDF-MST.

Verificamos que o algoritmo de benders para o MDF-MST não conseguiu superar o desempenho do algoritmo de plano de cortes em nenhuma das instâncias. Em média, o tempo

de computação foi 4 vezes maior. Na verdade, não executamos o algoritmo de benders nas instâncias maiores que as apresentadas na tabela, dado que já na última instância testada ele parou por exceder o limite máximo de tempo. O procedimeto que mais consome tempo computacional no benders é a resolução do problema mestre que cresce a medida que cortes são adicionados. Acreditamos que podemos melhorar esse tempo ao aplicar politicas de remoção de cortes de benders adicionadas ao mestre de tal forma que o problema mestre seja resolvido mais rápido sem com isso afetar a convergência do método.

Instâncias	Cplex T(s)	Benders T(s)
ALM60-1	1.28	2.04
ALM60-2	1.00	1.83
ALM60-3	0.94	5.53
ALM60-4	1.00	5.34
ALM100-1	11.96	23.05
ALM100-2	10.24	16.25
ALM100-3	9.40	43.01
ALM100-4	6.68	50.53
ALM200-1	328.68	459.83
ALM200-2	332.64	1768.30
ALM200-3	282.82	3498
ALM200-4	648.40	>9000
NEU60-1	1.26	1.84
NEU60-2	0.66	1.66
NEU60-3	1.02	7.93
NEU60-4	0.86	4.06
NEU100-1	13.80	24.72
NEU100-2	10.74	20.59
NEU100-3	8.72	75.59
NEU100-4	6.38	59.62
NEU200-1	527.50	1254.30
NEU200-2	446.84	964.88
NEU200-3	236.70	3241
NEU200-4	360.44	>9000

Tabela 5.18: Resultado da Decomposição de Benders para as instâncias das classes ALM e NEU

5.5 Heurística de Subgradientes

Neste Capítulo, iremos descrever um método heurístico que combina ingredientes da Decomposição de Benders (a forma como as variáveis são particionadas) e do Método de Subgradientes (a forma como o problema reformulado é resolvido) para gerar soluções viáveis de boa qualidade. Assim como na decomposição de benders, o subproblema escravo é dado pelo dual do problema resultante ao fixarmos os valores de um conjunto de variáveis no problema original. Esses valores são agora obtidos por iterações do método de subgradientes.

O método heurístico que propomos, desenvolvido a partir dessa ideia, é geral e pode ser aplicado tanto a problemas inteiros mistos quando inteiros puros. Na próxima seção, apresentamos os detalhes dessa heurística. Nas seções seguintes, apresentamos os resultados computacionais obtidos com sua aplicação aos problemas MDF-MST, DC-MST e MD-MST.

5.5.1 Definição da Heurística

Para facilitar a apresentação, consideramos primeiro o problema de programação mista:

$$
\begin{aligned}
P \quad & \min \quad d^T x + c^T y && (5.26)\\
& s.a: \quad Ax + By \geq b && (5.27)\\
& \qquad x \geq 0, y \in Y \subseteq \mathbb{Z}^q && (5.28)
\end{aligned}
$$

onde x e y são variáveis contínuas e inteiras, respectivamente, Y é o conjunto de pontos inteiros em um poliedro, A, B são matrizes e b, c e d são vetores de dimensões compatíveis.

Para cada $y \in Y$, definimos o subproblema paramétrico

$$
\begin{aligned}
P(y) \quad & \min \quad d^T x && \\
& s.a: \quad Ax \geq b - By && (5.29)\\
& \qquad x \geq 0 && (5.30)
\end{aligned}
$$

Para todo $y \in Y$, $P(y)$ é um problema linear. Seu dual é dado por:

$$
\begin{aligned}
D(y) \quad g(y) = \max \quad & \beta^T (b - By) && (5.31)\\
& s.a: \quad A^T \beta \leq d && (5.32)\\
& \qquad \beta \geq 0 && (5.33)
\end{aligned}
$$

onde consideramos $g(y) = +\infty$ se $D(y)$ é ilimitado. $D(y)$ será chamado de Subproblema da Heurística de Subgradientes.

Vamos admitir, daqui em diante, que $U = \{\beta \geq 0 : A^T \beta \leq d\}$ é não-vazio, ou seja, $D(y)$ é viável para todo y. Do contrário, pelo Teorema Fundamental da Dualidade Linear, para todo $y \in Y$, $P(y)$ é ilimitado ou inviável e, consequentemente, P não tem solução ótima.

Com essa hipótese, para todo $y \in Y$, temos que $g(y)$ descreve também o valor ótimo de $P(y)$, considerando que esse valor é $+\infty$ quando $D(y)$ é ilimitado e $P(y)$ inviável. Definindo,

$$f(y) = c^T y + g(y),$$

o problema P é reescrito como

$$P \quad \min f(y) \text{ s.a. } y \in Y.$$

Proposição 5.4 *f é convexa linear por partes.*

Prova. Temos que $f(y) = \sup\left\{c^T y + \beta^T (b - By) : \beta \in U\right\}$, ou seja, f é o supremo de uma família de funções lineares. Logo, f é conexa e linear por partes [47]. ■

Proposição 5.5 *Sejam $\bar{y} \in Y$ e $\beta_{\bar{y}}$ uma solução ótima de $D(\bar{y})$. Então, $s = c - B^T \beta_{\bar{y}}$ é um subgradiente de f em $\bar{y}$.*

Prova. Queremos mostrar que s satisfaz $f(y) - f(\bar{y}) \geq s^T (y - \bar{y})$, $\forall y \in Y$. Temos que $f(\bar{y}) = c^T \bar{y} + \beta_{\bar{y}}^T (b - B\bar{y})$ e $f(y) \geq c^T y + \beta_{\bar{y}}^T (b - By)$. Sendo assim,

$$\begin{aligned} f(y) - f(\bar{y}) &\geq c^T (y - \bar{y}) + \beta_{\bar{y}}^T \left[(b - By) - (b - B\bar{y})\right] \\ &= c^T (y - \bar{y}) - \beta_{\bar{y}}^T B(y - \bar{y}) \\ &= (c^T - \beta_{\bar{y}}^T B)(y - \bar{y}) \end{aligned}$$

Logo, s tal que $s = c - B^T \beta_{\bar{y}}$ é subgradiente de f em $\bar{y}$. ■

Em f sendo convexa e com subgradientes à mão, uma possibilidade para encontrar um limitante inferior para P é convexificar Y e aplicar o Método de Subgradientes Projetado. Mesmo que a convexificação ou a projeção sejam operações difíceis, a ideia de gerar pontos em Y seguindo a indicação de subgradientes pode ser interessante como heurística. Note que todo $y \in Y$ fornece um limite superior $f(y)$ para o problema P. Seguindo essa ideia, utilizando o método de subgradientes, queremos gerar sucessivos valores para y que retornem valores de f

próximos do ótimo. Abaixo, mostramos o algoritmo geral (para um problema de minimização).

Algoritmo 5.4: *Algoritmo da Heurística de Subgradientes para um Problema de Programação Mista*

Entrada: Parâmetros de Entrada: $(\pi, T_\pi, N, y^0, z_{UB})$

1. $k \leftarrow 0$
2. Enquanto condição de parada não for satisfeita, faça:
3. $\quad \beta^k \leftarrow resolve\_Dual(D(y^k))$;
4. $\quad f(y^k) \leftarrow c^t y^k + (\beta^k)^T (b - By^k)$;
5. $\quad$ Se $z_{UB} > f(y^k)$ faça:
6. $\quad\quad z_{UB} \leftarrow f(y^k)$;
7. $\quad\quad n_{stuck} \leftarrow 0$;
8. $\quad$ Caso contrário faça:
9. $\quad\quad n_{stuck} \leftarrow n_{stuck} + 1$;
10. $\quad$ Se $n_{stuck} = N$ faça:
11. $\quad\quad n_{stuck} \leftarrow 0$;
12. $\quad\quad \pi \leftarrow \varepsilon\pi$;
13. $\quad\quad$ Se $\pi \leq T_\pi$: PARE
14. $\quad s^k = c - B^T \beta^k$;
15. $\quad \alpha_k = \pi / \left\| s^k \right\|_2^2$;
16. $\quad \bar{y}^{k+1} = \bar{y}^k - \alpha_k s^k$;
17. $\quad$ Projete $\bar{y}^{k+1}$ em Y e obtenha $y^{k+1} \in Y$ de tal forma que $D(y^{k+1})$ seja limitado;
18. $\quad k \leftarrow k + 1$;
19. Retorne z_{UB};

Possivelmente o passo crucial no algoritmo acima seja o de número 17. Por projetar $\bar{y}^{k+1}$ sobre Y, entenda a determinação de um ponto $y^{k+1} \in Y$ muito próximo a $\bar{y}^{k+1}$. Essa operação pode ser realizada de várias formas, dependendo do conjunto Y. Ressaltamos que, ao projetarmos $\bar{y}^{k+1}$ em Y, devemos ter o cuidado de obter um valor para y^{k+1} de tal forma que $D(y^{k+1})$ não seja ilimitado.

Com relação aos parâmetros, eles podem ser escolhidos para cada problema específico, através de experimentos computacionais, assim como a atualização de π. No algoritmo acima, atualizamos $\pi = \varepsilon\pi$, onde $\varepsilon \in (0,1)$, sempre que N iterações do método são realizadas sem melhorar o melhor limite superior. Outros parâmetros podem ser utilizados para a finalização do algoritmo, como o número máximo de iterações e o tempo máximo permitido para o método.

Iremos mostrar agora que, mesmo quando o problema P é um problema inteiro puro, nosso algoritmo de subgradientes pode ser usado para encontrar soluções viáveis de P.

Considere o seguinte problema de programação inteira:

$$\begin{aligned} P^I \qquad \min \quad & d^T x + c^T y \\ s.a.: \quad & Ax + By \geq b \qquad (5.34) \\ & x \in \mathbb{Z}^p_+, y \in Y \subseteq \mathbb{Z}^q \qquad (5.35) \end{aligned}$$

Similarmente ao caso misto, para cada $y \in Y$, definimos:

$$\begin{aligned} P^I(y) \qquad g^I(y) = \min \quad & d^T x \\ s.a.: \quad & Ax \geq b - By \qquad (5.36) \\ & x \in \mathbb{Z}^p \qquad (5.37) \end{aligned}$$

Temos que $D(y)$, como definido em (5.31)–(5.33), também é um dual para $P^I(y)$, porém um dual fraco. Quer dizer, $D(y)$ fornece apenas um limite inferior para $P^I(y)$.

Para todo y, defina $f(y)$ como anteriormente e seja $f^I(y) = c^T y + g^I(y)$. Logo $f^I(y) \geq f(y)$, $\forall y \in Y$. Reescrevemos P^I como segue

$$P^I \quad \min f^I(y) \text{ s.a. } y \in Y.$$

Observamos que P é uma relaxação de P^I.

Proposição 5.6 *Sejam $\bar{y} \in Y$ e $\beta_{\bar{y}}$ uma solução ótima de $D(\bar{y})$. Então, $s = c - B^T \beta_{\bar{y}}$ é um $\varepsilon_{\bar{y}}$-subgradiente de f^I em $\bar{y}$, onde $\varepsilon_{\bar{y}} = f^I(\bar{y}) - f(\bar{y})$.*

Prova. Queremos mostrar que s satisfaz $f^I(y) - f^I(\bar{y}) \geq s^T(y - \bar{y}) - \varepsilon_{\bar{y}}$, $\forall y \in Y$. Temos que

$$\begin{aligned} f^I(\bar{y}) &= f(\bar{y}) + \varepsilon_{\bar{y}} = c^T \bar{y} + \beta_{\bar{y}}^T (b - B\bar{y}) + \varepsilon_{\bar{y}} \\ f^I(y) &\geq f(y) \geq c^T y + \beta_{\bar{y}}^T (b - By) \end{aligned}$$

Sendo assim,

$$\begin{aligned} f^I(y) - f^I(\bar{y}) &\geq c^T (y - \bar{y}) + \beta_{\bar{y}}^T [(b - By) - (b - B\bar{y})] - \varepsilon_{\bar{y}} \\ &= c^T (y - \bar{y}) - \beta_{\bar{y}}^T B(y - \bar{y}) - \varepsilon_{\bar{y}} \\ &= (c^T - \beta_{\bar{y}}^T B)(y - \bar{y}) - \varepsilon_{\bar{y}} \end{aligned}$$

Logo, s tal que $s = c - B^T \beta_{\bar{y}}$ é $\varepsilon_{\bar{y}}$-subgradiente de f em $\bar{y}$. ■

Baseados nesse resultado, propomos a seguir a versão da Heurística de Subgradientes para o caso inteiro puro. A diferença entre esta e a versão para o caso misto decorre do fato de que a solução de $D(y^k)$, um dual fraco, não fornece mais uma solução para $P^I(y^k)$. Para remediar tal incoveniente, iremos praticar a política de resolver tanto o problema dual $D(y)$

quanto o problema $P^I(y)$. Abaixo, mostramos o algoritmo geral.

Algoritmo 5.5: *Algoritmo da Heurística de Subgradientes para um Problema de Programação Inteira*

Entrada: Parâmetros de Entrada: $(\pi, T_\pi, N, y^0, z_{UB})$

1. $k \leftarrow 0$;
2. Enquanto condição de parada não for satisfeita, faça:
3. $\quad \beta^k \leftarrow resolve\_Dual(D(y^k))$;
4. $\quad x^k \leftarrow resolve\_Primal(P^I(y^k))$;
5. $\quad$ Se $z_{UB} > c^T y^k + d^T x^k$ faça:
6. $\quad\quad z_{UB} \leftarrow c^T y^k + d^T x^k$;
7. $\quad\quad n_{stuck} \leftarrow 0$;
8. $\quad$ Caso contrário faça:
9. $\quad\quad n_{stuck} \leftarrow n_{stuck} + 1$;
10. $\quad$ Se $n_{stuck} = N$ faça:
11. $\quad\quad n_{stuck} \leftarrow 0$;
12. $\quad\quad \pi \leftarrow \varepsilon\pi$;
13. $\quad\quad$ Se $\pi \leq T_\pi$: PARE
14. $\quad s = c - B^T \beta^k$;
15. $\quad \alpha_k = \pi / \left\| s^k \right\|_2^2$;
16. $\quad \bar{y}^{k+1} = \bar{y}^k - \alpha_k s^k$;
17. $\quad$ Projete $\bar{y}^{k+1}$ em Y e obtenha $y^{k+1} \in Y$ de tal forma que $D(y^{k+1})$ seja limitado;
18. $\quad k \leftarrow k + 1$;
19. Retorne z_{UB};

5.5.2 Heurística de Subgradientes para o MDF-MST

Utilizando qualquer das formulações apresentadas para MDF-MST no Capítulo 4, podemos desenvolver uma Heurística de Subgradientes. Escolhemos a formulação $\text{MDF}_{\text{subtour}}$, que possui um menor número de variáveis. Reescrevemos tal formulação abaixo.

$$(P) \quad \min \sum_{e\in\delta(C:T)} c_e x_e + \sum_{e\in E(C)} c_e y_e$$

$$\text{sujeito a: } \sum_{e\in E(C)} y_e = c-1 \tag{5.38}$$

$$\sum_{e\in E(S)} y_e \leq |S|-1, \forall S \subsetneq C \text{ com } |S|\geq 2 \tag{5.39}$$

$$\sum_{e\in\delta(i)\cap\delta(C:T)} x_e + \sum_{e\in\delta(i)\cap E(C)} y_e \geq d_i, \forall i \in C \tag{5.40}$$

$$\sum_{e\in\delta(t)\cap\delta(C:T)} x_e = 1, \forall t \in T \tag{5.41}$$

$$y_e \in \{0,1\}, \forall e \in E(C), x_e \geq 0, \forall e \in \delta(C:T) \tag{5.42}$$

Note que renomeamos as variáveis x_e, $\forall e \in E(C)$, por y_e, para ficar em conformidade com a notação usada em (5.26)–(5.28). Como vimos, a integralidade das variáveis x_e, $\forall e \in \delta(C:T)$, pode ser descartada, o que fazemos em (5.42). Sendo assim, a relaxação acima na verdade é equivalente ao modelo $\text{MDF}_{\text{subtour}}$ original com todas as variáveis inteiras.

Seja $Y = \{y \in \{0,1\}^{|E(C)|} : y \text{ satisfaz } (5.38)-(5.39)\}$, quer dizer o conjunto de árvores geradoras de $G(C)$. Para $y \in Y$, defina $D(y)$ como em (5.31)-(5.33), ou seja,

$$D(y) \quad g(y) = \max : \sum_{i\in C} \Delta_y(i) u_i - \sum_{t\in T} v_t - \sum_{\forall (i,j)\in\delta(C:T)} w_{ij} \tag{5.43}$$

$$\text{s. a.: } u_i - v_t - w_{it} \leq c_{it}, \forall (i,t) \in \delta(C:T) \tag{5.44}$$

$$u_i \geq 0, \forall i \in C,\ v_t \text{ livre}, \forall t \in T \text{ e } w_{it} \geq 0, \forall (i,t) \in \delta(C:T) \tag{5.45}$$

onde $\Delta_y(i) = d_i - \sum_{e\in\delta(i)\cap E(C)} y_e$ é a diferença entre o grau mínimo de i e seu grau na árvore descrita por y. Seja $f(y) = \sum_{e\in E(C)} c_e y_e + g(y)$. Como mostrado no início deste capítulo, $f(y)$ é conexa e linear por partes.

Dado $\bar{y} \in Y$, seja $(u(\bar{y}), v(\bar{y}), w(\bar{y}))$ uma solução ótima de $D(\bar{y})$. Temos que

$$g(\bar{y}) = \sum_{i\in C} \Delta_{\bar{y}}(i) u_i(\bar{y}) - \sum_{t\in T} v_t(\bar{y}) - \sum_{\forall (i,j)\in\delta(C:T)} w_{ij}(\bar{y})$$

Seguindo a prova da Proposição 5.5, obtemos um subgradiente de f em $\bar{y}$ como segue. Para qualquer $y \in Y$, temos

$$g(y) \geq \sum_{i\in C} \Delta_y(i) u_i(\bar{y}) - \sum_{t\in T} v_t(\bar{y}) - \sum_{\forall (i,j)\in\delta(C:T)} w_{ij}(\bar{y})$$

Note a diferença fundamental desta inequação em relação à equeção anterior, posto que passamos de $\Delta_{\bar{y}}$ para Δ_y. Então, chegamos a que

$$
\begin{aligned}
f(y) - f(\overline{y}) &= \sum_{e \in E(C)} c_e(y_e - \overline{y}_e) + g(y) - g(\overline{y}) \\
&\geq \sum_{e \in E(C)} c_e(y_e - \overline{y}_e) + \sum_{i \in C} \left[\Delta_y(i) - \Delta_{\overline{y}}(i)\right] u_i(\overline{y}) \\
&= \sum_{e \in E(C)} c_e(y_e - \overline{y}_e) - \sum_{i \in C} \left[\sum_{e \in \delta(i) \cap E(C)} (y_e - \overline{y}_e) \right] u_i(\overline{y}) \\
&= \sum_{e \in E(C)} c_e(y_e - \overline{y}_e) - \sum_{e=(i,j) \in E(C)} (y_e - \overline{y}_e)(u_i(\overline{y}) + u_j(\overline{y})) \\
&= \sum_{e=(i,j) \in E(C)} (c_{ij} - u_i(\overline{y}) - u_j(\overline{y}))(y_{ij} - \overline{y}_{ij})
\end{aligned}
$$

Logo s tal que $s_{ij} = c_{ij} - u_i(\overline{y}) - u_j(\overline{y})$ é subgradiente de f em $\overline{y}$.

A heurística de subgradientes para o MDF-MST considerando a formulação $\text{MDF}_{\text{subtour}}$ fica:

Algoritmo 5.6: *Heurística de Subgradientes para o MDF-MST*

Entrada: Parâmetros de Entrada: $(\pi, T_\pi, N, y^0, z_{UB})$

1. $k \leftarrow 0$;
2. Enquanto condição de parada não for satisfeita, faça:
3. $\quad (u,v)^k \leftarrow resolve\_Dual(D(y^k))$;
4. $\quad$ Se $z_{UB} > c^T y^k + g(y^k)$ faça:
5. $\quad\quad z_{UB} \leftarrow c^T y^k + g(y^k)$;
6. $\quad\quad n_{stuck} \leftarrow 0$;
7. $\quad$ Caso contrário faça:
8. $\quad\quad n_{stuck} \leftarrow n_{stuck} + 1$;
9. $\quad$ Se $n_{stuck} = N$ faça:
10. $\quad\quad n_{stuck} \leftarrow 0$;
11. $\quad\quad \pi \leftarrow \pi/2$;
12. $\quad\quad$ Se $\pi \leq T_\pi$: PARE
13. $\quad s^k_{ij} = c_{ij} - u^k_i - u^k_j, \forall (i,j) \in E(C)$;
14. $\quad \alpha_k = \pi / \left\| s^k \right\|^2_2$;
15. $\quad \overline{y}^{k+1} = \overline{y}^k - \alpha_k s^k$;
16. $\quad$ Projete $\overline{y}^{k+1}$ em Y e obtenha $y^{k+1} \in Y$;
17. $\quad k \leftarrow k + 1$;
18. Retorne z_{UB};

Lembre que Y descreve o conjunto de árvores geradoras de $G(C)$. A cada iteração, a projeção de $\overline{y}^{k+1}$ em Y para obter y^{k+1} será feita ordenando os valores de $\overline{y}^{k+1}$ em ordem

decrescente e aplicando a primeira etapa da Heurística Lagrangeana 3, o que garante que a árvore entre centrais projetada gere uma solução viável para o MDF-MST no subproblema.

Na linha 15 do Algoritmo 5.6, optamos por calcular o próximo iterado $\bar{y}^{k+1}$ a partir de $\bar{y}^k$ e não de y^k, pois depois de alguns experimentos computacionais verificamos que, quando utilizamos y^k, a heurística não realiza melhoras.

Aqui cabe uma observação. Desenvolvimento similar ao realizado acima para a formulação $\text{MDF}_{\text{subtour}}$ poderia ser feito para outras formulações como, por exemplo, $\text{MDF}^r_{\text{cutsetD}}$. Note que, neste caso, o conjunto Y seria como definido na Subseção 5.4.1. Embora também descreva as árvores geradoras de $G(C)$, Y agora está contido em um espaço de dimensão maior ($|A(C)|$ em lugar de $|E(C)|$). Além disso, um subgradiente em um ponto $\bar{y} \in Y$ seria dado por $s \in R^{|A(C)|}$ tal que $s_{ij} = c_{ij} - u_i(\bar{y})$, onde $u(\bar{y})$ é solução ótima de $D(\bar{y})$ definido em (5.17)–(5.19).

Repare que o dual $D(y)$ possui a variável w_{it} relacionado à restrição $y_{it} \geq 0$ do primal, diferentemente do dual da Seção 5.4.1. Resolvemos inserí-las por verificar que o algoritmo encontra melhores resultados quando comparado com o algoritmo sem essas variáveis com os mesmos valores de parâmetros, mas com tempo computacional um pouco pior.

5.5.2.1 Resultados Computacionais

A Heurística de Subgradientes inicia com uma árvore entre os centrais dada pela árvore da solução encontrada pela heurística gulosa baseada no PV1. Isto define o ponto y^0.

Os valores dos parâmetros foram $\pi = 8$, $T_\pi = 1$. O valor de N será dependente do tamanho do grafo, sendo $N = 120$ para instâncias com $c \leq 200$ e $N = 100$ para instâncias com $c \geq 300$. Esses valores foram obtidos após experimentos computacionais. O que podemos destacar é que para um valor alto de N para instâncias grande, acarretará no maior esforço computacional. Observamos que à medida que a solução encontrada se aproxima da solução ótima, ela precisa de uma maior quantidade de iterações para encontrar uma solução melhor, o que acaba acontecendo quando N é grande e que na maioria das vezes a diferença entre a duas soluções é insiginificante. Assim, o valor é reduzido para N, procurando estabelecer um compromisso entre a qualidade do limite superior e o tempo computacional exigido.

Nas Tabelas 5.19 e 5.20 encontramos a média e o desvio padrão do GAP. As instâncias que calculamos o GAP são instâncias para as quais conhecemos a solução ótima. Assim, o GAP é calculado como $(UB - w)/w$, onde UB é o limite superior e w a solução ótima. Comparamos a heurística de subgradientes com os resultados do PV1, VNS e heurística lagrangeana H2.

As soluções obtidas estão nas Tabelas 5.21 e 5.22. Podemos ver que o gap relativo médio fica abaixo de 0,5% para as intâncias ALM e abaixo de 2,7% para as NEU.

Comparando a heurística de subgradientes com PV1, observamos que há melhora em quase todas as soluções, exceto em 8 da classe ALM e em 21 da NEU. Observamos que para as instâncias maiores, a heurística de subgradiente não encontra solução melhor que a PV1. Para as instância ALM a partir de 500 centrais, observamos que para os grupos 3 e 4, ela

Algoritmos	Média GAP	Desvio Padrão GAP	Média Tempo (s)
Subgradiente	0,004495978	0,005863718	215,53
PV1	0,085080769	0,067418336	4,57
VNS	0,081181203	0,070637715	22,566875
Heurística H2	0,005019498	0,009858898	113,50

Tabela 5.19: Média e desvio padrão dos GAP e média dos tempos para as instâncias das classes ALM com ótimo conhecido

Algoritmos	Média GAP	Desvio Padrão GAP	Média Tempo (s)
Subgradiente	0,026971326	0,029307665	224,31
PV1	0,154767983	0,13027851	10,95
VNS	0,148477565	0,136253998	32,03
Heurística H2	0,015199073	0,022948172	168,02

Tabela 5.20: Média e desvio padrão dos GAP e média dos tempos para as instâncias das classes NEU com ótimo conhecido

sempre encontra solução melhor que o PV1 e para os grupos 1 e 2 isso não acontece.

Comparando com o VNS, verificamos que ela foi melhor em 53 casos (29 da ALM e 24 da NEU) e apenas em 11 da ALM e 16 da NEU ela não foi melhor. Como observamos na Seção 5.2, o VNS não encontra soluções melhores que o PV1 para os grupos 3 e 4 de todas as classes. Já a heurística de subgradientes consegue melhorar todas as soluções de PV1 para as instâncias desses grupos, exceto para as NEU a partir de 600 centrais. Na verdade, para a classe NEU com centrais a partir de 600, em apenas duas instâncias ele melhora o valor. Fazendo uma análise pelo gap relativo médio verificamos que heurística de subgradientes foi muito melhor em relação ao VNS.

Quando comparamos com a heurística H2, verificamos que para as ALM, em termos quantitativos, a H2 é melhor, encontrando soluções melhores em 26 instâncias contra 15 da heurística de subgradientes, mas quando analisamos o gap relativo médio e o desvio padrão médio, observamos que ela é melhor que a H2. Quando analisamos para as NEU, verificamos que a H2 obtem melhores soluções, seja em termos quantitativos, seja pelos valores do gap relativo médio e desvio padrão. Lembramos que esses valores estatísticos foram calculados apenas para as instâncias que os valores ótimos são conhecidos.

Com respeito aos tempos, verificamos que a heurística de subgradientes exige um esforço computacional equivalente a H2, quando calibramos os valores dos parâmetros. Observe que para as instâncias ALM a partir de 500 centrais dos grupos 3 e 4 a heurística parou por exceder o tempo máximo estabelecido. Como já haviamos dito, precisamos estabelecer um valor de N que tenha um compromisso entre encontrar uma boa solução e o tempo computacional. Acreditamos que para instâncias maiores que as testadas, precisamos que os parâmetros tenha outros valores, ainda menores, para que a heurística pare sem exceder o tempo máximo.

Instâncias	*w*	Subgradiente	T(s)	PV1	T(s)	VNS	T(s)	Heurística H2	T(s)
ALM60-1	6943	6945	3,46	7100	0,06	7042	0,74	**6943**	0,24
ALM60-2	6772	**6772**	2,76	7111	0,02	6962	0,83	**6772**	0,23
ALM60-3	6709	6711	3,21	7241	0,05	7219	0,34	**6709**	0,49
ALM60-4	7040	7041	1,97	7537	0,04	7537	0,35	**7040**	0,77
ALM100-1	8217	8226	13,43	8382	0,2	8305	2,62	**8217**	6
ALM100-2	8069	8070	11,31	8300	0,11	8248	2,3	**8069**	9,96
ALM100-3	8055	8062	16,66	9151	0,25	9151	1,12	**8055**	13,75
ALM100-4	8139	8147	10,91	9517	0,24	9517	1,04	**8139**	15,29
ALM200-1	11509	11541	96,59	11869	4,41	11829	22,99	**11509**	35,89
ALM200-2	11305	11334	147,7	11543	4,13	11532	30,25	**11317**	100,05
ALM200-3	11454	**11521**	213,19	13427	4,86	13427	10,85	11565	151,53
ALM200-4	11391	**11482**	215,1	13422	2,92	13422	8,72	11537	194,23
ALM300-1	13462	13498	452,96	13749	14,96	13733	87,81	**13464**	334,84
ALM300-2	13004	13191	153,1	13283	13,5	13188	121,18	**13004**	258,46
ALM300-3	12797	**12992**	1221,64	14900	17,48	14900	38,25	13154	425,86
ALM300-4	12545	**12754**	884,53	14736	9,84	14736	31,68	12905	268,48
ALM400-1		15740	1563,51	16060	39,02	15967	312,46	**15526**	955,18
ALM400-2		15567	444,17	15595	44,17	15541	384,73	**15117**	827,98
ALM400-3		**15657**	3752	18606	60,92	18606	120,5	15659	1658,91
ALM400-4	15255	**15377**	2656	18390	50,61	18390	107,26	15495	1737,85
ALM500-1		17875	1280,22	17875	77,7	17830	794,36	**17328**	2054,62
ALM500-2		17732	1222,03	17750	118,68	17690	800,49	**17162**	2493,12
ALM500-3		**17339**	9037	19804	159,16	19804	280,64	17360	2109,52
ALM500-4		**18016**	9032	21114	163,51	21114	286,76	18063	3601,52
ALM600-1		19523	2736	19553	218,79	19525	1559,99	**19012**	7735,19
ALM600-2		19392	2334	19392	247,23	19304	1723,09	**18652**	7045,84
ALM600-3		19893	9059	22892	314,44	22892	553,07	**19865**	8230,73
ALM600-4		19749	9035	23500	184,93	23500	422,26	**19699**	8414,54
ALM700-1		20974	4567	20974	462,69	20956	2322,86	**20417**	9049,19
ALM700-2		20513	4070	20513	448,3	20466	2583,07	**19949**	8986,43
ALM700-3		**21597**	9094	24330	623,09	24330	1855,8	21894	9220,05
ALM700-4		**21479**	9090	24953	515,99	24953	1979,55	22152	9123,36
ALM800-1		22214	3745	22214	749,17	22156	4104,85	**21618**	9180,35
ALM800-2		22066	3533	22066	715,16	22039	4919,72	**21345**	9227,3
ALM800-3		**24606**	9200	27406	1232,14	27406	3233,98	25365	9503,54
ALM800-4		**27498**	9176	30386	1060,08	30386	3353,85	28745	9538,82
ALM900-1		23293	5285	23293	1203,69	23260	6245,09	**22856**	9620,99
ALM900-2		22691	5027	22691	1247,95	22623	6752,54	**22276**	9457,03
ALM900-3		**28639**	9286	31989	1786,89	31989	5148,43	30285	9997,07
ALM900-4		**28042**	9255	31700	1454,55	31700	5216,32	29545	10172

Tabela 5.21: Resultado da heurística de subgradientes para as instâncias das classes ALM

Instâncias	*w*	Subgradiente	T(s)	PV1	T(s)	VNS	T(s)	Heurística H2	T(s)
NEU60-1	3032	**3032**	2,94	3149	0,03	3084	0,71	**3032**	0,68
NEU60-2	3872	**3872**	4,44	4060	0,16	3985	0,85	**3872**	0,91
NEU60-3	2972	3001	4,13	3661	0,03	3661	0,28	**2973**	2,55
NEU60-4	4031	4037	4,58	4889	0,1	4889	0,4	**4031**	1,1
NEU100-1	3079	3122	7,36	3132	0,37	3124	2,94	**3079**	5,32
NEU100-2	3858	3943	17,29	4018	0,67	3992	5,34	**3859**	10,6
NEU100-3	3084	**3107**	29,46	3936	0,24	3936	1,01	3152	7,74
NEU100-4	3655	**3666**	25,14	4453	0,63	4453	1,54	3699	13,75
NEU200-1	3326	3484	43,82	3484	4,21	3406	35,87	**3326**	39,93
NEU200-2	4280	4420	61,54	4420	12,9	4388	35,52	**4284**	116,04
NEU200-3	3388	**3533**	416,91	4235	3,17	4235	9,2	3579	74,3
NEU200-4	4223	**4264**	707,48	5837	8,29	5837	14,86	4398	158,02
NEU300-1	3462	3564	135,23	3564	15,79	3531	117,63	**3464**	476,9
NEU300-2	4382	4568	253,46	4568	72,2	4524	183,69	**4382**	648,94
NEU300-3	3326	**3518**	1629,64	4573	14,11	4573	36,67	3532	538,12
NEU300-4	4001	4447	245,64	4929	42,25	4929	65,93	**4190**	593,37
NEU400-1		3652	507,56	3652	72,48	3630	388,5	**3549**	1330
NEU400-2		4372	967,61	4372	154,96	4355	468,54	**4222**	2290,31
NEU400-3		4303	297,38	4568	55,06	4568	114,05	**3826**	1320,19
NEU400-4		5103	670,64	5212	148,12	5212	209,38	**4338**	1738,5
NEU500-1		3641	1336,13	3641	138,1	3594	743,86	**3513**	4188,37
NEU500-2		4525	2301	4525	420,07	4499	1104,17	**4366**	3405,68
NEU500-3		4607	832,34	4756	153,79	4756	278,39	**3930**	3998,53
NEU500-4		5083	2268	5978	311,02	5978	443,33	**4632**	3282,61
NEU600-1		3779	2715	3779	285,02	3765	1900,52	**3676**	5851,4
NEU600-2		4709	4557	4709	692,55	4686	2443,96	**4554**	6899,62
NEU600-3		4672	1759	4824	249,86	4824	483,82	**4064**	4854,18
NEU600-4		5545	3100	5545	563,27	5545	806,45	**4620**	9225,2
NEU700-1		3972	4715	3972	517,56	3932	2948,87	**3823**	9180,44
NEU700-2		4906	8096	4906	1332,4	4867	3351,92	**4766**	9337,32
NEU700-3		5134	2760	5134	459,76	4959	2226,73	**4507**	9147,93
NEU700-4		5741	5497	5741	1085,62	5741	2566,41	**5002**	9482,83
NEU800-1		4084	5071	4084	806,98	4051	4753,12	**3962**	9205,33
NEU800-2		5103	5358	5103	2257,66	5063	6401,2	**4950**	10142,78
NEU800-3		5129	3566	5753	679,73	5753	3684,29	**4466**	9326,72
NEU800-4		5854	3954	5854	1799,73	5854	4254,38	**5603**	9768,61
NEU900-1		4297	5548	4297	1111,5	4267	6734,51	**4186**	9843,66
NEU900-2		5197	7631	5197	2934,8	5176	9167,32	**5070**	11264,88
NEU900-3		**5018**	3706	5018	1249,76	5161	6147,17	5175	9490,93
NEU900-4		6171	5745	6171	2393,63	6171	6264,19	**5446**	10545,63

Tabela 5.22: Resultado da heurística de subgradientes para as instâncias das classes NEU

5.6 Conclusões

Neste Capítulo, propormos vários algoritmos para o MDF-MST. Concluirmos que para instâncias de até 300 vértices centrais, o algoritmo de Plano de Corte com CPLEX mostrou-se ser o mais eficiente, encontrando a solução ótima em tempo computacional aceitável. O mesmo não podemos dizer para as instâncias maiores de 300 vértices centrais. O mesmo algoritmo não encontra nem a relaxação linear, logo, nem mesmo uma solução viável. Já os algoritmos lagrangeano, mostraram-se eficientes para todas as instâncias, encontradno boas soluções viáveis com tempos computacionais aceitáveis. Lembrando que por de trás dos algoritmos lagrangeanos temos heurísticas lagrangeanas e a relaxação lagrangeana para encontrar limites superioes e inferiores, respectivamente. Em alguns casos, o gap lagrangeano garante a otimalidade da solução encontrada. As heurísticas lagrangeanas sempre encontram melhores soluções viáveis em comparação com os demais métodos heurísticas apresentado neste capítulo, mesmo comparando com o algoritmo VNS, conhecido na literatura por encontrar soluções viáveis de boa qualidade. Verificamos também que a Heurística de Subgradiente, apesar de ter um embasamento teórico, ou seja, mistura ingredientes da decomposição de benders com o método de subgradiente, não mostrou-se ser melhor que as heurísticas lagrangeanas. Quando a comparamos com os demais algoritmos heurísticos, verificamos que ela é competitiva.

6 DC-MST - ÁRVORE GERADORA MÍNIMA COM RESTRIÇÃO DE GRAU MÁXIMO

Seja $G = (V,E)$ um grafo conexo e não direcionado onde, a cada vértice $v \in V$, vamos associar um inteiro positivo d_v. O *Problema da Árvore Geradora com Restrição de Grau* (DC-ST – *Degree Constrained Spanning Tree*) consiste em encontrar uma árvore geradora de G tal que o grau de cada vértice v nessa árvore seja menor ou igual a d_v. Garey e Johnson [26] provaram que esse problema é NP-Completo.

O DC-ST é um problema de decisão, já que queremos encontrar qualquer solução viável. Quando consideramos um grafo ponderado e desejamos encontrar uma árvore geradora mínima que respeite as restrições de grau máximo dos vértices, estamos diante do *Problema da Árvore Geradora Mínima com Restrição de Grau* (DC-MST). Mais precisamente, conforme definição apresentada na Seção 3.1, associado um custo $c_e \in \mathbb{R}$ a cada aresta $e \in E$, o DC-MST consiste em encontrar uma árvore geradora mínima de G tal que o grau de cada vértice v nessa árvore seja menor ou igual a d_v.

O DC-MST é um problema NP - Difícil [26], cuja definição em muito se assemelha a de MDF-MST. Tal semelhança se traduz diretamente nas formulações matemáticas para os dois problemas. Mais do que isso, mostramos neste capítulo que boa parte dos algoritmos que propomos aqui para o MDF-MST podem ser facilmente adaptados para DC-MST, especialmente para resolver instâncias onde há vários vértices v com $d_v = 1$. Detalhamos especialmente a aplicação do algoritmo lagrangeano, que se mostrou o mais efetivo para o MDF-MST.

6.1 Formulações Matemáticas

A formulação matemática para o DC-MST que tem sido utilizada em praticamente todos os artigos na literatura é baseada no conjunto de restrições de eliminação de ciclos (*Subtour Elimination Constrainsts*, SECs) adicionado à restrição de grau máximo.

$$
\begin{aligned}
(\mathrm{DC}_{\mathrm{subtour}}) \quad & \min \sum_{e \in E} c_e x_e \\
& \text{sujeito a: } \sum_{e \in E(C)} x_e = |V| - 1 \\
& \sum_{e \in E(S)} x_e \leq |S| - 1, \forall S \subsetneq V \text{ com } |S| \geq 2 \\
& \sum_{e \in \delta(v)} x_e \leq d_v, \forall v \in V \\
& x_e \in \{0,1\}, \forall e \in E
\end{aligned}
$$

Esta é a formulação usada, por exemplo, em [44],[12],[15],[7],[16],[17],[5] e [6].

Cunha e Lucena [17] apresentaram uma desigualdade válida para o problema. Ela

advém de uma desigualdade indutora de faceta para o politopo do *u-Capacitated b-Matching* para o caso $u = 1$ [22]. Sejam $S \subseteq V$, $J \subseteq \delta(S)$ e $d(S) = \sum_{i \in S} d_i$. Em toda solução viável para DC-MST, temos:

$$\sum_{e \in E(S)} x_e + \sum_{e \in J} x_e \leq \left\lfloor \frac{1}{2}(|S|\, d + |J|) \right\rfloor, \forall S \subseteq V, \forall J \subseteq \delta(S). \tag{6.1}$$

Tal desigualdade é não-reduntante quando $(d(S) + |J|)$ é ímpar.

Uma outra formulação que pode ser utilizada para o DC-MST á baseada nas restrições de cortes direcionados (*Directed Cutset Constraints*, DCUTs) adicionada à restrição de grau máximo. Como de costume, iremos fazer a transformação do grafo não direcionado $G = (V, E)$ no grafo direcionado $D = (V, A)$. Seja a variável binária y_{ij} que irá indicar se o arco (i, j) estará ou não na solução e seja uma raiz $r \in V$. Assim obtemos a seguinte formulação para o DC-MST.

$$\begin{aligned}
(\mathrm{DC}^r_{\mathrm{cutsetD}}) \ \min & \sum_{(i,j) \in A} c_{ij} y_{ij} \\
\text{sujeito a:} & \sum_{(i,j) \in A} y_{ij} = |V| - 1 \\
& \sum_{(i,j) \in \delta^+(S)} y_{ij} \geq 1, \forall S \subsetneq V \text{ e } r \in S \\
& \sum_{(i,j) \in \delta^+(i)} y_{ij} \leq d_i - 1, \forall i \in V \backslash \{r\} \\
& \sum_{(i,j) \in \delta^+(r)} y_{ij} \leq d_r \\
& \sum_{(j,i) \in \delta^-(i)} y_{ji} = 1, \forall i \in V \backslash \{r\} \\
& y_{ij} \in \{0, 1\}, \forall (i, j) \in A
\end{aligned}$$

Essa formulação não é muito utilizada na literatura para o DC-MST.

6.1.1 Simplificações nos Modelos

O DC-MST está fortemente relacionado ao MDF-MST, e isso pode ser claramente percebido comparando suas formulações. Muito mais que uma simples semelhança das formulações, vamos mostrar que podemos aplicar ao DC-MST boa parte dos algoritmos obtidos aqui para o MDF-MST.

Para isso, similarmente ao MDF-MST, vamos também particionar os vértices do grafo de entrada $G = (V, E)$ de DC-MST em $V = C \cup T$, onde $T = \{v \in V : d_v = 1\}$ e $C = V \setminus T$. Repare que os vértices em T são obrigatoriamente folhas, enquanto os vértices em C estão livres para serem folhas ou vértices internos. Embora em geral possamos ter $T = \emptyset$, encontramos na literatura muitas instâncias difíceis com uma boa quantidade de vértices em T. Vamos tirar proveito desse fato para simplificar as formulações $\mathrm{DC}_{\mathrm{subtour}}$ e $\mathrm{DC}^r_{\mathrm{cutsetD}}$.

Baseados nas formulações $\mathrm{MDF}_{\mathrm{subtour}}$ e $\mathrm{MDF}^r_{\mathrm{cutsetD}}$, e pelos bons resultados obtidos

por elas, vamos propor formulações para o DC-MST que consideram os vértices terminais. Assim, temos:

$$(\mathrm{DCT}_{\mathrm{subtour}}) \quad \min \sum_{e\in E} c_e x_e$$

$$\text{sujeito a:} \sum_{e\in E(C)} x_e = c-1 \tag{6.2}$$

$$\sum_{e\in E(S)} x_e \leq |S|-1, \forall S \subsetneq C \text{ com } |S| \geq 2 \tag{6.3}$$

$$\sum_{e\in \delta(v)} x_e \leq d_v, \forall v \in C \tag{6.4}$$

$$\sum_{e\in \delta(C)\cap\delta(t)} x_e = 1, \forall t \in T \tag{6.5}$$

$$x_e \in \{0,1\}, \forall e \in E \tag{6.6}$$

$$(\mathrm{DCT}^r_{\mathrm{cutsetD}}) \quad \min \sum_{(i,j)\in A} c_{ij} y_{ij}$$

$$\text{sujeito a:} \sum_{(i,j)\in A(C)} y_{ij} = c-1 \tag{6.7}$$

$$\sum_{(i,j)\in \delta^+(S:C\setminus S)} y_{ij} \geq 1, \forall S \subseteq C \text{ e } r \in S \tag{6.8}$$

$$\sum_{(i,j)\in \delta^+(i)} y_{ij} \leq d_v - 1, \forall i \in C\setminus\{r\} \tag{6.9}$$

$$\sum_{(i,j)\in \delta^+(r)} y_{ij} \leq d_r \tag{6.10}$$

$$\sum_{(i,j)\in \delta^-(t)\cap\delta(C:T)} y_{ij} = 1, \forall t \in T \tag{6.11}$$

$$\sum_{(j,i)\in \delta^-(i)} y_{ji} = 1, \forall i \in C\setminus\{r\} \tag{6.12}$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{6.13}$$

Note que em (6.2)–(6.3) e (6.7)–(6.8) consideramos as restrições de cardinalidade e SEC's/DCUT's apenas em $G(C)$, diferentemente de $\mathrm{DC}_{\mathrm{subtour}}$ e $\mathrm{DC}^r_{\mathrm{cutsetD}}$, onde tais restrições são descritas em G. Isto leva a uma simplificação do modelo. Note ainda que em (6.9) não colocamos restrições referentes a $i \in T$, que na verdade correspondem a anular as variáveis y_{ij}, para todo $i \in T$ e todo $(i,j) \in \delta^+(i)$.

6.1.2 Avaliação Computacional

Com a finalidade de avaliarmos as duas (re)formulações apresentadas, resolvemos fazer uma comparação com as formulações originais para o DC-MST. Para isso, vamos utilizar

o algoritmo proposto neste trabalho, na Seção 5.1, para o MDF-MST, modificado para o DC-MST. Para o $DC^r_{cutsetD}$ e o $DCT^r_{cutsetD}$, o algoritmo começa com o algoritmo de planos de corte para resolver a relaxação linear, utilizando política de inserção e remoção de restrições DCUT's. Quando o algorimo resolve a relaxação linear, ele adiciona as restrições de integralidade e passa a resolver o problema inteiro, que inicia com as restrições de corte da última iteração da relaxação linear. Para o $DC_{subtour}$ e o $DCT_{subtour}$, o algoritmo de planos de corte não utiliza nenhuma política de remoção de restrições.

Diferentemente do que fizemos com o MDF-MST, implementamos os algoritmos para as duas formulações. Isto porque a formulação baseada em SECs é usada recorrentemnete na literatura para o DC-MST, enquanto a formulação baseada em DCUTs produziu bons resultados para o MDF-MST.

Na literatura, temos dois grupos de instâncias para o DC-MST, introduzidas por Cunha e Lucena [17], chamadas de DE e DR. As instâncias DE são geradas no plano euclideano com restrições de grau variando entre 1, 2 e 3. Já as instâncias DR são instâncias não euclideanas com custos obtidos de forma aleatória, a partir de uma distribuição uniforme no intervalo $[1, 1000]$. Mais recentemente, novas instâncias euclideanas e não-eucideanas para o DC-MST foram obtidas em [11]. Não temos informações adicionais sobre essas novas instâncias, mas percebemos que elas possuem boa quantidade de vértices em T. Iremos chamá-las de instâncias LH.

Nas Tabelas 6.1 a 6.4, apresentamos os resultados obtidos para as formulações. Para cada instância, o valor na coluna c é a quantidade de vértices em C e na coluna w temos a solução ótima. As colunas "DCT-DCUT", "DCT-SEC", "DC-DCUT"e "DC-SEC"contêm, respectivamente, o tempo das formulações $DCT^r_{cutsetD}$, $DCT_{subtour}$, $DC^r_{cutsetD}$ e $DC_{subtour}$, sendo que a raiz r selecionada é a mesma.

Comparando cada formulação de acordo com seu tipo de restrições, verificamos que as re-formulações são bem mais eficientes do que as originais. A economia de tempo chega a ser até de cinco vezes em algumas instância DE/DR e 18 vezes para uma instância LH. Isto possibilitou inclusive resolver otimamente um maior número de instâncias. O destaque em negrito na Tabela 6.1 identifica valores ótimos que não eram conhecidos para as instâncias DE, conforme informação obtida em Cunha e Lucena [17] e Freitas [24]. Vale ressaltar, como desde o início relatamos, que essa reformulação é mais eficiente quando trabalhamos com instâncias com boa quantidade de terminais.

Para algumas instâncias, os valores dos tempos estão com "-", significando que não executamos o algoritmo para tal instância. Isso porque na última instância executada, seguindo a sequência de tamanho, ocorreu falta de memória; concluímos que o mesmo iria acontecer com as instâncias maiores.

Para as instâncias LH, inserirmos uma nova coluna, w_{RL}, com o valor da relaxação linear. No caso em que a coluna do tempo está preenchido e a coluna w_{RL} vazia, significa que a relaxação linear é inteira.

Fazendo uma comparação geral, os resultados nos mostram que a re-formulação

baseada nas DCUTs é muito mais eficiente do que a re-formulação baseada em SECs. O ganho da primeira em relação ao ganho da segunda chegou a ser seis vezes maior em uma instâncias DE. Observe que a $DCT^{r}_{cutsetD}$ resolveu instâncias com até 500 vértices de todos os grupos e uma de 600 para o grupo DE; já a $DCT_{Subtour}$ resolveu até 400 vértices.

Instâncias	c	w	DCT-DCUT	DC-DCUT	DCT-SEC	DC-SEC
DE 100_1	64	8779	0.89	3.28	0.40	1,3
DE 100_2	65	9629	0.66	3.10	0.81	2,35
DE 100_3	64	10798	1.48	6.42	2.23	7,15
DE 200_1	140	12031	33.20	94.83	19.98	76,66
DE 200_2	128	**12645**	28.91	109.61	40.41	154,89
DE 200_3	137	12229	36.17	135.90	39.34	186,48
DE 300_1	211	**14792**	248.27	1134.91	300.56	1229,36
DE 300_2	215	13931	184.69	592.46	453.74	1189,45
DE 300_3	199	**14997**	256.32	905.54	344.74	1182,07
DE 400_1	266	19239	208.82	1043.79	1476.41	sem memória
DE 400_2	263	**17419**	895.62	3915	2065,45	-
DE 400_3	292	16091	1032	3472	3189	-
DE 500_1	347	**19357**	1823.98	sem memória	sem memória	-
DE 500_2	333	**22400**	2316	sem memória	-	-
DE 500_3	344	**19411**	2995.50	sem memória	-	-
DE 600_1	404	**21930**	5032	-	-	-
DE 600_2	419	-	sem memória	-	-	-
DE 600_3	402		sem memória	-	-	-

Tabela 6.1: Comparação entre as formulações do DC-MST para as instâncias da classe DE

Instâncias	c	w	DCT-DCUT	DC-DCUT	DCT-SEC	DC-SEC
DR 100_1	64	2186	0.86	3.54	1.68	4.43
DR 100_2	65	2674	1.38	5.72	0.33	2.98
DR 100_3	64	2689	0.82	4.56	0.43	3.69
DR 200_1	140	2141	29.18	148.14	51.46	202.72
DR 200_2	128	2471	36.30	141.82	28.81	124.50
DR 200_3	137	2405	24.21	109.89	22.33	159.61
DR 300_1	211	2462	309.04	937.95	308.31	2365
DR 300_2	215	2312	311.70	1191.72	232.64	1136.28
DR 300_3	199	2585	265.05	1195.09	326.23	1625.51
DR 400_1	266	2817	599.78	sem memória	sem memória	sem memória
DR 400_2	263	2443	821.29	sem memória	-	-
DR 400_3	292	2387	2044.21	sem memória	-	-
DR 500_1	347	2597	2801	-	-	-
DR 500_2	333	2652	2607	-	-	-
DR 500_3	344	2593	3333	-	-	-
DR 600_1	404	2820	sem memória	-	-	-
DR 600_2	419	2820	sem memória	-	-	-
DR 600_3	402	2800	sem memória	-	-	-

Tabela 6.2: Comparação entre as formulações do DC-MST para as instâncias da classe DR

Instâncias	c	w_{RL}	w	DCT-DCUT	DC-DCUT	DCT-SEC	DC-SEC
LHE 100_3	64	11618	11640	1.82	6.32	2.70	5.89
LHE 100_4	61		11748	0.39	2.30	3.05	9.01
LHE 100_5	53	12340	12345	0.82	6.23	1.21	4.95
LHE 200_3	132	15158,5	15177	20.62	77.06	83.11	302.87
LHE 200_4	117	14657,5	14671	14.41	94.90	51.11	168.37
LHE 200_5	102	14942,5	14967	9.54	60.81	21.34	155.87
LHE 300_3	202	15631	15691	247.95	402.80	586.96	484.44
LHE 300_4	175	19717	19731	89.22	759.90	359.15	1882.39
LHE 300_5	164	18486,5	18492	82.75	1018.98	238.49	1644.48
LHE 400_3	273	20438.50	20454	347.01	1495.23	sem memória	sem memória
LHE 400_4	229	21004.00	21005	280.79	2885	-	-
LHE 400_5	209	24907.00	24911	453.96	sem memória	-	-
LHE 500_3	336	20544.08	20584	3134	-	-	-
LHE 500_4	291	27107.00	27124	2376	-	-	-
LHE 500_5	268		27389	1053.33	-	-	-
LHE 600_3	401	-	-	sem memória	-	-	-
LHE 600_4	345	-	-	-	-	-	-
LHE 600_5	325	-	-	-	-	-	-
LHE 700_3	460	-	-	-	-	-	-
LHE 700_4	409	-	-	-	-	-	-
LHE 700_5	378	-	-	-	-	-	-

Tabela 6.3: Comparação entre as formulações do DC-MST para as instâncias da classe LH Euclideano

Instâncias	c	w_{RL}	w	DCT-DCUT	DC-DCUT	DCT-SEC	DC-SEC
LHN 100_3	68	2140,38	2149	3.91	7.16	2.67	4.99
LHN 100_4	60		2327	0.48	3.78	0.44	1.99
LHN 100_5	52	2411	2419	1.40	4.32	0.60	3.09
LHN 200_3	134	3106	3108	34.67	154.97	31.98	133.00
LHN 200_4	114		3144	6.26	71.11	10.51	66.23
LHN 200_5	113	3335	3336	16.38	201.40	20.63	117.04
LHN 300_3	195	4110	4110	123.49	867.09	66.71	1204.28
LHN 300_4	169	4205	4206	81.26	984.46	137.72	1435.91
LHN 300_5	164	4342	4342	108.14	1769.75	150.06	1934.30
LHN 400_3	265	5113.79	5117	608.46	sem memória	1809.59	sem memória
LHN 400_4	239	5232.50	5233	575.99	-	836.58	-
LHN 400_5	217	5354.60	5356	445.35	-	708.96	-
LHN 500_3	328	5997.88	5999	3889	-	sem memória	-
LHN 500_4	288	6193.00	6194	2044.57	-	-	-
LHN 500_5	275	6167.36	6168	1610.05	-	-	-
LHN 600_3	400	-	-	sem memória	-	-	-
LHN 600_4	343	-	-	-	-	-	-
LHN 600_5	322	-	-	-	-	-	-
LHN 700_3	470	-	-	-	-	-	-
LHN 700_4	409	-	-	-	-	-	-
LHN 700_5	368	-	-	-	-	-	-

Tabela 6.4: Comparação entre as formulações do DC-MST para as instâncias da classe LH Não-Euclideano

6.2 Relaxação Lagrangeana

6.2.1 Algoritmos Lagrangeanos

Devido aos bons resultados apresentados para o MDF-MST pelos algoritmos que usam a relaxação lagrangeana, seja pelo limite superior da heurística lagrangeana ou pelo limite inferior do dual lagrangeano, concentramos nossos esforços em adaptá-los para o DC-MST.

Optamos por usar a re-formulação de DC-MST dada por restrições SECs. Todo o desenvolvimento relacionado à relaxação lagrangeana para essa formulação é praticamente igual àquele feito na Seção 5.3 para o MDF-MST. As únicas diferenças estão na função do problema lagrangeano,

$$\psi(x,u) = \sum_{e=(i,j)\in E(C)} x_e(c_e + u_i + u_j) + \sum_{e=(i,t)\in\delta(C:T)} x_e(c_e + u_i) + \sum_{i\in C} u_i d_i.$$

e na forma como se calcula o sugradiente,

$$s_i^k = \sum_{e\in\delta(i)} x_e - d_i$$

Também utilizamos o método de subgradientes utlizando a variação DSM para encontrar os valores dos multiplicadores que maximizam o valor do dual lagrangeano para o DC-MST, da mesma forma como aplicado ao MDF-MST.

Igualmente ao que fizemos para o MDF-MST, desenvolvemos duas heurísticas lagrangeanas para o DC-MST, uma correspondente à Heurística H1 e outra à Heurística H2. A primeira utiliza o Procedimento de Viabilidade PV1, considerando os custos originais das arestas para realizar as trocas e encontrar uma solução viável a partir da solução do problema lagrangeano. A segunda é similar, mas considera os custos lagrangeanos em lugar dos custos originais.

Destacamos que o Procedimento de Viabilidade precisou ser modificado para ser aplicado ao DC-MST. Praticamente a única alteração foi que, ao invés de aumentar o grau dos vértices centrais que não satisfazem sua restrição de grau, o PV1 para o DC-MST decrementa o grau dos vértices centrais que não satisfazem sua restrição de grau.

6.2.2 Resultados Computacionais

Agora iremos fazer uma comparação das nossas heurísticas lagrangeanas com os resultados de Cunha e Lucena [17] e de Freitas [24] para as instâncias DE e DR. Para as instâncias LH, iremos fazer apenas a comparação entre as nossas duas heurísticas lagrangeanas, uma vez que não dispomos de resultados computacionais para as mesmas na literatura.

Nas Tabelas 6.5 e 6.6 encontram-se os valores dos limites superiores obtidos pelos

algoritmos para as instâncias DE e DR. Verificamos a boa qualidade obtida pelos algoritmos de Cunha e Lucena [17] e Freitas [24]. Observamos que nossas heurísticas lagrangeanas encontram bons limites superiores, valores bem próximos da solução ótima.

Destacamos que nossas heurísticas são algoritmos simples, de fácil implementação, que usam uma relaxação lagrangeana de uma formulação simplificada e o Procedimento de Viabilidade PV1. Diferentemente dos algoritmos tomados como base para comparação, nossos algoritmos não utilizam estratégias elaboradas para encontrar os limites superiores. Eles usufruem basicamente da estrutura desta formulação simplificada baseada na formulação do MDF-MST, que se mostra assim efetiva, motivando-nos a continuar trabalhando com a mesma.

O método de sugradientes implementado para o DC-MST é igual ao do MDF-MST, ou seja, o Algoritmo 5.3. Os multiplicadores lagrangeanos iniciais e os parâmetros também são os mesmos, ou seja, fizemos $u_v = 1/|C|$, $\forall v \in C$, e $\pi = 2$, $T_\pi = 0.1$, $N = 50$ e $\lambda = 0.05$.

Fazendo uma análise apenas entre as nossas duas heurísticas lagrangeanas, verificamos que a Heurística H2 foi mais eficiente em boa parte das instâncias. Podemos constatar isso pelos gap e desvio padrão relativos médio. Em termos quantitativos, de um total de 78 instâncias, a H2 foi melhor em 40, a H1 em 22 e em 16 obtiveram o mesmo valor. Quando comparamos dentro de um mesmo grupo, verificamos que para as instâncias DE/DR ambas as heurísticas são equivalentes, enquanto que para as instâncias LH a H2 é superior, encontrando melhores soluções em 26 contra 8 da H1, de um total de 42 instâncias.

Todos os algoritmos foram executados em ambiente computacional diferentes, exceto as nossas duas heurística lagrangeana, de modo que não podemos comparar os tempos. Eles estão sendo exibidos apenas para passar uma ideia da ordem de grandeza.

Instâncias	w	H1	T(s)	H2	T(s)	Freitas	T(s)	Cunha	T(s)
DE 100_1	8779	**8779**	8,26	**8779**	8,92	**8779**	50	**8779**	5
DE 100_2	9629	**9629**	8,93	**9629**	5,13	**9629**	171	**9629**	4
DE 100_3	10798	10822	6,21	10800	4,7	**10798**	1413	**10798**	18
DE 200_1	12031	**12031**	198,01	**12031**	219,35	**12031**	3766	**12031**	54
DE 200_2	12645	12753	86,19	12709	101,5	12665	388800	**12645**	193
DE 200_3	12229	12345	169,36	12353	174,24	**12229**	3147	**12229**	101
DE 300_1	14792	14878	875,25	14867	748,14	14796	388800	**14792**	260
DE 300_2	13931	13954	1387,42	13955	1303,27	**13931**	103904	**13931**	89
DE 300_3	14997	15062	606,71	15076	556,66	15009	388800	**14997**	310
DE 400_1	19239	19675	800,58	19675	759,69	**19239**	355773	**19239**	989
DE 400_2	17419	17576	1587,05	17589	1003,56	17479	388800	**17419**	1580
DE 400_3	16091	16112	3878,59	16108	4418,72	**16091**	388800	**16091**	584
DE 500_1	19357	19503	3822,72	19509	3342,99	19445	388800	**19357**	794
DE 500_2	22400	23272	2476,38	22839	3543,14	22754	388800	**22400**	3558
DE 500_3	19411	19729	3291,41	19682	2780,02	19493	388800	**19411**	7811
DE 600_1	21930	22092	6159,61	22182	3995,94	22039	388800	**21930**	7763
DE 600_2	-	21796	5730,78	21711	6304,98	21663	388800	**21449**	11966
DE 600_3	-	22624	7209,53	22705	5617,63	22368	388800	**22243**	5303
Média GAP/Tempo(s):		0,00846	2127,94	0,00712	1938,25	0,00223	263612	0	2299
Desvio Padrão GAP:		0,01024		0,00711		0,00407		0	

Tabela 6.5: Resultados das Heurísticas Lagrangeanas do DC-MST para as instâncias da classe DE

Instâncias	*w*	H1	T(s)	H2	T(s)	Freitas	T(s)	Cunha	T(s)
DR 100_1	2186	**2186**	8,17	**2186**	8,81	**2186**	24	**2186**	8
DR 100_2	2674	**2674**	3,79	**2674**	3,51	**2674**	2	**2674**	6
DR 100_3	2689	**2689**	7,9	**2689**	4,18	**2689**	18	**2689**	8
DR 200_1	2141	2185	221,05	2181	187,94	**2141**	45	**2141**	61
DR 200_2	2471	**2471**	82,88	2506	54,87	**2471**	117	**2471**	74
DR 200_3	2405	2451	120,64	2455	132,7	**2405**	101	**2405**	62
DR 300_1	2462	2494	854,99	2509	741,24	**2462**	241	**2462**	189
DR 300_2	2312	2324	1204,81	2317	1257,24	**2312**	212	**2312**	107
DR 300_3	2585	2744	409,22	2716	467,83	**2585**	311	**2585**	423
DR 400_1	2817	3102	752,93	3140	547,17	**2817**	4259	**2817**	1395
DR 400_2	2443	2599	892,16	2559	1046,04	**2443**	4557	**2443**	713
DR 400_3	2387	2429	2866,14	2436	3141,53	**2387**	381	**2387**	161
DR 500_1	2597	2674	3725,28	2684	4056,74	**2597**	3622	**2597**	1007
DR 500_2	2652	2921	1652,56	2826	1932,73	**2652**	3481	**2652**	5233
DR 500_3	2593	2692	4624,15	2691	4402,23	**2593**	2089	**2593**	1098
DR 600_1	2820	3135	3945,07	3027	5020,32	**2820**	4933	**2820**	2160
DR 600_2	2820	2914	6345,04	2864	5295,91	**2820**	41611	**2820**	8256
DR 600_3	2800	3058	5331,83	2958	6282,4	**2800**	13002	**2800**	3296
Média GAP/Tempo(s):		0,03936	1836,03	0,03281	1921,29	0	4389,22	0	1347,61
Desvio Padrão GAP:		0,03927		0,03071		0		0	

Tabela 6.6: Resultados das Heurísticas Lagrangeanas do DC-MST para as instâncias da classe DR

Instâncias	*w*	Heurística H1	T(s)	Heurística H2	T(s)
LHE 100_3	11640	**11661**	5.57	**11661**	3.80
LHE 100_4	11748	**11748**	2.64	**11748**	4.34
LHE 100_5	12345	12346	2.31	**12345**	1.86
LHE 200_3	15177	15186	66.97	**15183**	47.20
LHE 200_4	14671	**14671**	61.45	14681	56.87
LHE 200_5	14967	**14974**	32.07	**14974**	32.97
LHE 300_3	15691	16510	234.49	**16194**	245.11
LHE 300_4	19731	20287	212.31	**20078**	240.23
LHE 300_5	18492	**18492**	222.73	18547	213.39
LHE 400_3	20454	21184	595.25	**20776**	760.51
LHE 400_4	21005	21426	850.53	**21178**	893.68
LHE 400_5	24911	25461	807.02	**25402**	839.36
LHE 500_3	20584	21584	1816.48	**21346**	1577.94
LHE 500_4	27124	27587	2179.17	**27498**	2287.59
LHE 500_5	27389	28131	1285.82	**27691**	1340.75
LHE 600_3	-	25406	5263.79	**25267**	5224.65
LHE 600_4	-	28281	4418.46	**28137**	4094.17
LHE 600_5	-	**25516**	2152.56	25604	2038.27
LHE 700_3	-	27618	8774.81	**27179**	8074.31
LHE 700_4	-	29649	5366.20	**29360**	5629.92
LHE 700_5	-	32578	7319.78	**32264**	7618.05
Média GAP/Tempo:		0,016924933	1984,30	0,010765933	1963,09
Desvio Padrão GAP:		0,018465931	-	0,011944935	-

Tabela 6.7: Resultados das Heurísticas Lagrangeanas do DC-MST para as instâncias da classe LH Euclideano

Instâncias	*w*	Heurística H1	T(s)	Heurística H2	T(s)
LHN 100_3	2149	2197	5.46	**2160**	6.35
LHN 100_4	2327	**2327**	3.39	**2327**	4.50
LHN 100_5	2419	**2419**	3.48	**2419**	2.69
LHN 200_3	3108	3356	51.74	**3149**	72.62
LHN 200_4	3144	**3144**	28.33	**3144**	28.04
LHN 200_5	3336	**3356**	39.66	**3356**	28.26
LHN 300_3	4110	4213	208.51	**4163**	179.33
LHN 300_4	4206	**4306**	148.54	4322	134.38
LHN 300_5	4342	4463	151.35	**4410**	137.48
LHN 400_3	5117	5323	544.74	**5280**	495.64
LHN 400_4	5233	5410	387.72	**5372**	511.49
LHN 400_5	5356	**5453**	343.63	5518	309.88
LHN 500_3	5999	6280	1159.65	**6171**	1474.51
LHN 500_4	6194	6395	1050.80	**6364**	1041.31
LHN 500_5	6168	6296	766.68	**6289**	770.56
LHN 600_3	-	7187	2559.00	**7133**	3535.82
LHN 600_4	-	**7320**	2115.86	**7320**	2078.96
LHN 600_5	-	**7435**	1540.92	7451	1273.89
LHN 700_3	-	**8257**	4801.52	8285	4496.95
LHN 700_4	-	8386	3717.28	**8382**	4043.51
LHN 700_5	-	**8288**	2214.36	8307	2412.24
Média GAP/Tempo:		0,025137697	1040,12	0,016322642	1097,06
Desvio Padrão GAP:		0,020982702	-	0,011973828	-

Tabela 6.8: Resultados das Heurísticas Lagrangeanas do DC-MST para as instâncias da classe LH Não-Euclideano

6.3 Heurística de Subgradientes para o DC-MST

Também implementamos a heurística de subgradientes apresentada na Seção 5.5 para o DC-MST, usando a formulação $\text{DCT}_{\text{subtour}}$. Praticamente toda a teoria é a mesma que a desenvolvida para o MDF-MST, exceto pelo dual $D(y)$ que terá o domínio das variáveis $u_i \leq 0$. Por isso, omitimos aqui os detalhes desse desenvolvimento.

Abaixo temos o algoritmo utilizado para o DC-MST.

Algoritmo 6.1: *Heurística de Subgradientes para o DC-MST*

Entrada: Parâmetros de Entrada: $(\pi, T_\pi, N, y^0, z_{UB})$

1. $k \leftarrow 0$;
2. Enquanto condição de parada não for satisfeita, faça:
3. $\quad (u,v,w)^k \leftarrow resolve\_Dual(D(y^k))$;
4. $\quad$ Se $z_{UB} > c^T y^k + g(u,v,w)^k$ faça:
5. $\quad\quad z_{UB} \leftarrow c^T y^k + g(u,v,w)^k$;
6. $\quad\quad n_{stuck} \leftarrow 0$;
7. $\quad$ Caso contrário faça:
8. $\quad\quad n_{stuck} \leftarrow n_{stuck} + 1$;
9. $\quad$ Se $n_{stuck} = N$ faça:
10. $\quad\quad n_{stuck} \leftarrow 0$;
11. $\quad\quad \pi \leftarrow \pi/2$;
12. $\quad\quad$ Se $\pi \leq T_\pi$: PARE
13. $\quad s_{ij}^k = c_{ij} - u_i^k, \forall (i,j) \in A(C)$;
14. $\quad \alpha_k = \pi / \left\| s^k \right\|_2^2$;
15. $\quad \bar{y}^{k+1} = \bar{y}^k - \alpha_k s^k$;
16. $\quad$ Projete $\bar{y}^{k+1}$ em Y e obtenha $y^{k+1} \in Y$;
17. $\quad k \leftarrow k+1$;
18. Retorne z_{UB};

6.3.1 Resultados Computacionais

A forma como realizamos a projeção constituiu-se na simples aplicação de um Kruskal modificado, considerando os valores de $\bar{y}^{k+1}$ em ordem não decrescente. Nesse Kruskal modificado, uma aresta só é incluída na árvore se não formar ciclo e se não for incidente a um vértice cujo grau máximo já tenha sido atingido. No final, ele garante que a árvore entre centrais encontrada pode originar uma solução viável para o DC-MST no subproblema.

Os parâmetros utilizados no algoritmo foram: $\pi = 16$, $T_\pi = 1$ e $N = 220$.

Na tabelas 6.10 e 6.11 apresentamos os resultados computacionais com a heurística

de subgradientes para as instâncias das classes DE, DR e LH. Percebemos que ela é capaz de encontrar boas soluções, especialmente para as classes DE e LH euclideano, onde o gap relativo médio fica abaixo de 0,8%. Na tabela 6.9 fazemos uma comparação entre o gap relativo médio dos resultados de Cunha e Lucena [17], de Freitas [24] e a heurística H2 para as instâncias DE e DR. Já para as instâncias LH comparamos apenas entre nossas duas heurísticas.

Entretanto, esses resultados não se comparam favoravelmente com aqueles apresentados na Seção 6.2, quando comparamos com os resultados de Cunha e Lucena [17] e de Freitas [24]. Agora, quando comparamos com heurística lagrangeana H2, verificamos que para as instâncias LH ela possui um gap e desvio padrão relativo médio melhor que a heurística H2, o mesmo já não acontece para as instâncias DE e DR. Em termos quantitativos, de 15 instâncias testadas de cada grupo, a heurística de subgradiente encontra solução melhor em 4 das DE, 4 das DR, 8 das LHE e 8 das LHN. Com relação aos tempos computacionais, a heurística de subgradiente é mais eficiente, como podemos observar pela média dos tempos na tabela 6.9.

Instâncias	Algoritmos	Média GAP	Desvio Padrão GAP	Média Tempo (s)
DE	H. Subgradientes	0,008901787	0,006523335	43,53
	Heurística H2	0,007114271	0,007107461	1938,25
	Freitas	0,002227578	0,004071499	-
	Cunha	0	0	-
DR	H. Subgradientes	0,053262088	0,0214888	40,34
	Heurística H2	0,032809955	0,030713746	1921,29
	Freitas	0	0	-
	Cunha	0	0	-
LHE	H. Subgradientes	0,006212785	0,00736318	271,46
	Heurística H2	0,010765933	0,011944935	1963,09
LHN	H. Subgradientes	0,015988965	0,010973131	559,89
	Heurística H2	0,016322642	0,011973828	1097,06

Tabela 6.9: Média e desvio padrão dos GAP e média dos tempos para as instâncias das classes DE, DR e LH

Instâncias	w	H. Subgradientes	T(s)	Instâncias	w	H. Subgradiente	T(s)
DE 100_1	8779	8785	3,03	DR 100_1	2186	2192	3,16
DE 100_2	9629	9833	7,37	DR 100_2	2674	2752	6,7
DE 100_3	10798	10807	5,07	DR 100_3	2689	2787	2,88
DE 200_1	12031	12124	29,87	DR 200_1	2141	2296	19,39
DE 200_2	12645	12729	31,05	DR 200_2	2471	2562	57,61
DE 200_3	12229	12244	21,78	DR 200_3	2405	2556	20,98
DE 300_1	14792	14881	82,55	DR 300_1	2462	2590	91,77
DE 300_2	13931	14116	56,88	DR 300_2	2312	2474	61,34
DE 300_3	14997	15092	154,17	DR 300_3	2585	2697	99,22
DE 400_1	19239	19614	376,09	DR 400_1	2817	3003	663,55
DE 400_2	17419	17597	246,35	DR 400_2	2443	2558	281,53
DE 400_3	16091	16168	146,9	DR 400_3	2387	2576	182,93
DE 500_1	19357	19702	505,14	DR 500_1	2597	2816	433,77
DE 500_2	22400	22570	997,61	DR 500_2	2652	2820	1625,64
DE 500_3	19411	19599	463,36	DR 500_3	2593	2731	564,19

Tabela 6.10: Resultado da Heurística de Subgradientes para as instâncias das classes DE e DR

Instâncias	*w*	H. Subgradientes	T(s)	Instâncias	*w*	H. Subgradiente	T(s)
LHE 100_3	11640	11716	4.62	LHN 100_3	2149	2210	5.33
LHE 100_4	11748	11853	3.32	LHN 100_4	2327	2377	5.61
LHE 100_5	12345	12348	2.50	LHN 100_5	2419	2427	3.76
LHE 200_3	15177	15286	28.23	LHN 200_3	3108	3222	33.95
LHE 200_4	14671	14713	37.26	LHN 200_4	3144	3168	71.71
LHE 200_5	14967	15004	40.74	LHN 200_5	3336	3374	57.89
LHE 300_3	15691	16071	203.29	LHN 300_3	4110	4161	288.30
LHE 300_4	19731	19760	137.91	LHN 300_4	4206	4273	166.05
LHE 300_5	18492	18529	104.83	LHN 300_5	4342	4371	134.64
LHE 400_3	-	20763	388.80	LHN 400_3	-	5225	766.70
LHE 400_4	-	21249	339.46	LHN 400_4	-	5302	714.46
LHE 400_5	-	25086	303.18	LHN 400_5	-	5414	696.86
LHE 500_3	-	21116	1176.89	LHN 500_3	-	6078	1999.88
LHE 500_4	-	27405	785.21	LHN 500_4	-	6257	1708.74
LHE 500_5	-	27514	515.62	LHN 500_5	-	6221	1744.56

Tabela 6.11: Resultado da Heurística de Subgradientes para as instâncias da classe LH

7 CONCLUSÕES E TRABALHOS FUTUROS

Nesta tese, apresentamos uma variação do Problema de Árvore Geradora Mínima com Restrição de Grau Mínimo (Min-Degree Constrained Minimum Spannig Tree - MD-MST). O MD-MST consiste em encontrar uma árvore geradora mínima de um grafo onde cada vértice ou é folha da árvore ou satisfaz uma restrição de grau mínimo. Os vértices folhas são chamados terminais e os demais são os centrais. Na variação do MD-MST apresentada aqui, que chamamos de Problema da Árvore Geradora Mínima com Restrição de Grau Mínima e Centrais e Terminais Fixos (MDF-MST), os terminais e centrais são definidos a priori.

Realizamos um estudo da complexidade do problema. Mostramos que decidir se existe ou não uma solução viável para o MDF-MST é um problema NP-Completo, mesmo que existam no grafo todas as possíveis arestas entre centrais e terminais. Apresentamos também várias propriedades relacionando a quantidade de centrais e terminais com viabilidade do problema. Como consequência da NP-Completude do problema de viabilidade, concluímos que o MDF-MST é NP-Difícil. Provamos também que este problema está na classe FPT, parametrizado pelo número de centrais. Identificamos casos em que o problema torna-se polinomial, o que ocorre em particular quando o subgrafo induzido pelos centrais é uma árvore. Neste caso, uma solução ótima pode ser encontrada resolvendo um problema de fluxo de custo mínimo.

Propomos várias formulações de programação inteira para o problema, desde formulações baseadas nas tradicionais restrições de corte (*Directed Cutset Constraints*, DCUTs) ou nas restrições de eliminação de ciclos (*Subtour Elimination Constrainsts*, SECs), que possuem um grupo de restrições em quantidade exponencial, como também formulações compactas. Nesse segundo grupo, temos as formulações com restrições de conectividade via conservação de fluxo, via coincidência de arestas (segundo Martin [37]) ou baseadas nas restrições Miller-Tucker-Zemlin [42] (segundo Gouveia [28]).

Relacionamos teoricamente os limites inferiores gerados pelas relaxações lineares dessas formulações. Comparamos também computacionalmente a qualidade desses limites e o seu tempo de computação, através de um algoritmo de planos de corte (para as formulações via DCUTs e SECs) e resolução direta pelo CPLEX (para as formulações compactas). Para o caso específico do nosso algoritmo de planos de corte para a formulação baseada em DCUTs, utilizamos, além da tradicional política de adição de restrições violadas, políticas de remoção de restrições inativas. O algoritmo com essa política de remoção de restrições inativas foi notadamente mais eficiente do que o algoritmo sem ela. Verificamos a excelente qualidade dos limites das relaxações lineares das formulações baseadas em DCUTs, SECs e Coincidência de Arestas para as instâncias da classe ALM, geradas por Martins e Souza [38], Em muitos casos, a própria solução linear já era inteira, garantindo que ela também é a solução ótima do problema inteiro.

Após isso, procuramos testar a eficiência do solver CPLEX em resolver, através das formulações propostas, o problema inteiro. Nosso algoritmo inicia com o algoritmo de planos de corte para resolver a relaxação linear; caso ela seja inteira, o algoritmo retorna tal solução; caso contrário, adiciona as restrições de integralidade e continua o processo de planos de corte.

A formulação escolhida foi a baseada em DCUTs, devido à qualidade do limite inferior que gera. Utilizamos as instâncias ALM da literatura, propostas para o MD-MST, onde apenas tivemos que definir quem são os centrais. Tal algoritmo mostrou-se ser bastante eficiente, resolvendo todas essas instâncias em um tempo bastante aceitável.

Procuramos então gerar instâncias potencialmente mais difíceis, variando parâmetros que se mostraram influentes na complexidade do problema. De fato, quando executamos o nosso algoritmo de plano de corte para essas novas instâncias, o mesmo não conseguiu encontrar soluções ótimas ou mesmo resolver a relaxação linear em boa parte dos casos, tendo havido falta de memória ou atingido o limite máximo de tempo de 9000s. Dessas novas instâncias geradas, metade são euclideanas, que também chamamos ALM, e metade não-euclideanas, que chamamos NEU. As instâncias de cada um dos dois tipos estão divididas em 4 grupos, conforme a dificuldade esperada.

Desejando encontrar soluções viáveis de boa qualidade, desenvolvemos várias heurísticas gulosas baseadas em trocas de aresta e heurísticas baseadas em busca em vizinhança. Os Procedimentos de Viabilidades que definimos geram soluções viáveis a partir de árvores geradoras que não respeitam algumas restrições de grau.

Definimos o Problema da Árvore Geradora Mínima com Terminais Fixos (MST-F), que ocorre quando descartamos em MDF-MST as restrições de grau dos centrais. Mostramos um algoritmo polinomial que encontra sua solução ótima. Utilizamos a solução ótima do MST-F como entrada de três Procedimentos de Viabilidade, que se diferenciam basicamente pela ordem considerada para as trocas de arestas. Os três se mostraram capazes de encontrar boas soluções, e não detectamos um que seja notadamente mais eficiente que os demais.

Além disso, definimos uma busca VND e uma heurística VNS. O nosso algoritmo VNS recebe como entrada a solução encontrada pelo Procedimento de Viablidade 1 (PV1). Verificamos que o VNS encontra soluções melhores em praticamente todas as instâncias dos grupos 1 e 2, mas em nehuma dos grupo 3 e 4. Esses dois grupos, 3 e 4, são aqueles onde as instâncias satisfazem as condições da Proposição 3.6 na igualdade.

Dada a boa qualidade do valor da relaxação linear para as formulações, mas alto custo para resolvê-la, propomos e testamos uma relaxação lagrangeana da formulação via SECs, que usamos também para definir heurísticas lagrangenas. O problema lagrangeano resultante da dualização das restrições de grau para a função objetivo é um MST-F. Nossas heurísticas lagrangeanas praticamente são a aplicação do Procedimento de Vaibilidade 1, recebendo como entrada a solução do problema lagrangeano a cada iteração do método de subgradientes. A ordem em que as trocas de arestas são realizadas considera os custos originais ou aqueles modificados pelos multiplicadores lagrangeanos. Verificamos que essa segunda estratégia gera melhores resultados. Comparando os resultados dos algoritmos lagrangeanos com aqueles do solver CPLEX, verificamos a eficiência computacional dos primeiros, sendo comparável ou superior em muitos casos ao do solver.

Realizamos uma reformulação para o MDF-MST baseada nas restrições DCUTs de tal moda que aplicamos o método de benders ao problema. A idéia é visualizar o MDF-MST como a composição de duas variáveis onde na primeira se decide quem são as arestas entre

os centrais e a segunda as arestas dos centrais para os terminais de tal modo que as restrições de grau sejam satisfeitas. Nosso objetivo era encontra melhores resultados comparado com o nosso algoritmo de plano de corte que utiliza o CPLEX.

O Problema da Árvore Geradora Mínima com Restrição de Grau (DC-MST) assemelha-se ao MDF-MST por procurar também uma árvore geradora mínima satisfazendo uma restrição de grau para cada vértice, só que neste caso uma uma restrição de grau *máximo*. Muitas das instâncias disponíveis na literatura para o DC-MST possuem uma boa quantidade de vértices com restrição de grau máximo igual a 1, ou seja, esses vértices obrigatoriamente serão folhas na solução. Dessa forma, similarmente ao MDF-MST, visualizamos o DC-MST com o conjunto de vértices particionado entre os terminais (com restrição de grau máximo igual a 1) e centrais (restrição de grau máximo estritamente maior que 1). A partir disso, adaptamos para o problema DC-MST boa parte das formulações e algoritmos propostos para o MDF-MST.

Primeiramente, apresentamos duas novas formulações, baseadas nas formulações do MDF-MST, onde as restrições de árvores são aplicadas somente aos vértices centrais. A comparação realizada, utilizando o solver CPLEX, entre as novas formulações e as correspondentes formulações tradicionais usadas na literatura mostrou que aquelas são significativamente mais eficientes. Em seguida, adaptamos para o DC-MST os algoritmos lagrangeanos propostos para o MDF-MST, usando a nova formulação baseada em SECs. Comparando os resultados das heurísticas lagrangeanas com os trabalhos da literatura, concluímos que nossas heurísticas lagrangeanas são competitivas e promissoras, isso levando em conta que essas heurísticas lagrangeanas são algoritmos simples que exploram apenas a estrutura da nova formulação e não utilizam nenhum método mais sofisticado para encontrar soluções de melhor qualidade, como ocorre nos trabalhos da literatura.

Por fim, propomos uma nova heurística geral que combina ingredientes da Decomposição de Benders com Método de Subgradientes, a qual denominamos Heurística de Subgradientes. Os resultados obtidos por essa heurística para o MDF-MST, DC-MST e MD-MST, apesar de não serem os melhores, comparados com os resultados já existentes, mostram-nos que ela é promissora e pode ser melhorada, principalmente em se tratando da forma como "projetamos". Basicamente, a cada iteração, geramos um ponto com entradas reais que desejamos converter num vetor binário que descreva uma árvore geradora (no caso do MDF-MST e DC-MST) ou o conjunto de vértices centrais (no caso do MD-MST). A forma como fazemos essa conversão é a mais simples possível, devido ao pouco tempo que tínhamos para finalizar esse trabalho. Apenas ordenamos as componentes do ponto relaxado corrente e selecionamos, as arestas ou vértices, conforme o problema, na ordem não-crescente desses valores.

Há ainda algumas questões que, futuramente, gostaríamos de estudar, com respeito ao trabalho realizado. Uma delas é procurar melhorar os resultados do algoritmo VNS para o MDF-MST, principalmente para os grupos 3 e 4 de instâncias. Para isso desejamos implementar um algoritmo de segunda ordem, introduzido por Karnaugh [33] e utilizado em Martins e Souza [38] com bons resultados para o DC-MST. Um algoritmo de Segunda Ordem (SO) gera uma sequência de problemas restritos que se relacionam. Ele inicia uma iteração com um problema restrito P; depois adiciona em P uma restrição para gerar um novo problema restrito Q, tcujo conjunto viável esteja contido no de P. Para problemas de árvore geradora, as restrições que

geram os problemas restritos são restrições que fixam arestas. Dessa forma, ele fixa uma aresta em P para gerar o problema Q.

Todas as nossas heurísticas gulosas para o MDF-MST apresentadas neste trabalho são baseadas na transformação da solução ótima do MST-F em uma solução de boa qualidade do MDF-MST. Desejamos desenvolver heurísticas gulosas que gerem suas próprias soluções viáveis de boa qualidade para o MDF-MST sem serem baseadas na solução do MST-F.

Pelos resultados mostrados aqui neste trabalho, verificamos que, para instâncias grandes, os algoritmos apresentam um custo computacional excessivo. Podemos citar os algoritmos lagrangeanos, que frequentemente param após um pequeno número de iterações, pelo critério de tempo máximo (e não o de convergência, como desejado), quando aplicados às instâncias de maior porte. Pensando nisso e devido aos bons resultados de Andrade et al.[7], desejamos utilizar a ideia de trabalhar com um grafo reduzido, gerado a partir de uma solução viável. A ideia é não trabalhar com todas as arestas do grafo mas com apenas um subconjunto dessas. Uma forma de determinar esse grafo reduzido seria aplicar o algoritmo de Kruskal para encontrar a árvore geradora mínima e manter nesse grafo reduzido todas as arestas do grafo original com custo menor ou igual ao custo da aresta de maior custo na árvore geradora juntamente com um percentual extra de arestas.

Outra forma possível para melhorar o desempenho das Heurísticas Lagrangeanas para instâncias maiores é desenvolver um Procedimento de Viabilidade menos oneroso que, mesmo não encontrando boas soluções quando aplicado isoladamente, leva a bons bons resultados quando incorporado a Heurística Lagrangeana e aplicado iterativamente. Constatamos após alguns experimentos computacionais que o PV é o que mais consome tempo nos algoritmos lagrangeano. O Algoritmo Lagrangeano 3 foi uma tentativa de atingir tal objetivo, mas que não logrou o êxito desejado. Já para as instâncias menores onde não fechamos o gap de otimalidade, podemos utilizar parâmetros mais agressivos que resultem em maior quantidade de iterações. Podemos citar, por exemplo, um valor maior para o parâmetro N. Isso tanto para o MDF-MST quanto para o DC-MST, já que em ambos as heurísticas lagrangeanas são bastante promissoras. Em outras palavras, uma calibragem mais cuidadosa dos parâmetros pode contribuir para a obtenção de resultados ainda melhores das Heurísticas Lagrangeanas. Também desejamos desenvolver um algoritmo lagrangeano que obtem apenas o limite inferior, ou seja, um algoritmo lagrangeano sem uma heurística lagrangeana.

Um dos aspectos relevantes para a eficiência de um algoritmo Branch and Bound é a boa qualidade do limite inferior utilizado. Pelos resultados apresentados para as formulações do MDF-MST, verificamos a boa qualidade dos valores da relaxação linear, mas com significativo custo computacional. Por outro lado, verificamos os bons resultados do algoritmo lagrangeano, seja pela sua heurística, com bons limites superiores, seja pelos limites inferiores, que sabemos que é no máximo o valor da relaxação linear. Dados esses bons resultados, iniciamos o desenvolvimento de um algoritmo Branch and Bound para o MDF-MST que utiliza a relaxação lagrangeana para obter o limite inferior e a heurística lagrangeana para obter o limite superior. Estamos utilizando como regra de ramificação uma adaptação da estratégia introduzida recentemente por Freitas [24], chamada de árvore de subgradientes. Pelos resultados preliminares, observamos que o limite inferior juntamente com o superior realizam várias podas nos

ramos, impedindo que a árvore de ramificação cresça. Poucos nós são efetivamente explorados. Constatamos, porém, nesses testes preliminares um alto tempo de computação para garantir a solução ótima. Acreditamos que se deve ao fato de estarmos utilizando um Procedimento de Viabilidade com a heurística lagrangeana. Como relatamos anteriormente, o PV é o processo que mais consome tempo no algorimto lagrangeano. Como os resultados são inconclusivos, decidimos não inserir no conteúdo deste trabalho, deixando apenas como trabalho futuro.

Para a Heurística de Subgradientes, desejamos estudar novas formas de fazer a projeção realizada a cada iteração. Ao invés de fazer uma ordenação simples dos valores, podemos utilizar alguma informação dos custos do problema para direcionar a projeção, seja de uma forma gulosa ou utilizando outro critério mais elaborado. Desejamos expandir a ideia de Andrade et al. [7] para o MDF-MST e DC-MST, assim como fizemos para o MD-MST.

No algoritmo de Benders para o MDF-MST, verificamos que a resolução do problema mestre é o procedimento que mais consome tempo computacional no algoritmo, com isso, desejamos criar politicas de remoção de cortes de benders de tal modo que seja mais rápido resolver o problema mestre e não interfira na convergência do método.

Também desejamos desenvolver um algoritmo Local Branching para o MDF-MST e para o DC-MST baseado nas formulações apresentada neste trabalho.

Finalmente, dados os resultados efetivos dos algoritmos Lagrangeanos e um desempenho razoável da Decomposição de Benders para o MDF-MST, acreditamos que um algoritmo do tipo Cross Decomposition pode resultar em uma boa alternativa de solução. A Cross Decomposition é um método desenvolvido por Van Roy [50], que unifica a Decomposição de Benders e a Relaxação Lagrangeana.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] **Achuthan, N.; Caccetta, L.; Caccetta, P.; Geelen, J**. *Computational methods for the diameter restricted minimum weight spanning tree problem*. Australasian Journal of Combinatorics, v. 10, p. 51-71, 1994.

[2] **Ahuja, R. K.; Magnanti, T. L.; Orlin J.B**. *Network flows: Theory, algorithms and applications.* New Jersey: Prentice-Hall; 1993.

[3] **Almeida, A. M.; Martins, P.; Souza, M.C**. *Min-Degree Constrained Minimum Spanning Tree Problem: Complexity, proprieties and formulations.* International Transactions in Operational Research, v. 19, p. 323-352, 2012.

[4] **Akgún, I.; Tansel, B. C.** *Min-degree constrained minimum spanning tree problem: new formulation via Miller-Tucker-Zemlin constraints.* Computers and Operations Research, v. 37(1), p. 72-82, 2009.

[5] **Andrade, R. C.; Freitas, A.T**. *Otimização em árvore de subgradiente para a árvore geradora mínima com restrição de grau nos vértices*. In: XL Simpósium Brasileiro de Pesquisa Operacional, 2008, João Pessoa-PB. XL SBPO, 2008. p. 1751-1759.

[6] **Andrade, R. C.; Freitas, A.T**. Disjunctive combinatorial branch in a subgradient tree algorithm for the DCMST problem with VNS-Lagrangian bounds. Electronic Notes in Discrete Mathematics, v. 41, p. 5-12, 2013.

[7] **Andrade, R. C.; Lucena, A.; Maculan, N**. *Using lagrangian dual information to generate degree constrained spanning trees*. Discrete Appl. Math. v. 154 (5), p. 703–717, 2006.

[8] **Barahona, F.; Anbil, R**. *The volume algorithm: producing primal solutions with subgradient method*. Mathematical Programming, v. 87, p. 385–399, 2000.

[9] **Beasley, J. E**. Lagrangean relaxation, em Reeves, C. R. (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, p. 243-303, 1993.

[10] **Benders, J. F**. *Partitioning procedures for solving mixed-variables programming problems*. Numerische Mathematik, v. 4(1), p. 238-252, 1962.

[11] **Bicalho, L.H.; Cunha, A.S. e Lucena, A**. *Instâncias de Teste para o DC-MST*. Disponível em: <http://homepages.dcc.ufmg.br/~luishbicalho/>, Acesso em: 20/04/2014.

[12] **Caccetta, L.; Hill, S.P.** *A Branch and cut method for the degree constrained minimum spanning tree problem*. Networks, v. 37(2), p. 74-83, 2001.

[13] **Camerini, P.; Fratta, L.; Maffioli, F**. *On improving relaxation methods by modified gradient techniques*. Mathematical Programming Study, v. 3, p. 26–34, 1975.

[14] **Conforti, M.; Cornuéjols, G¦ Zambelli, G.**. *Extended Formulation in Combinatorial Optimization*. Annals of Operations Research, v. 204, p. 97-143, 2013.

[15] **Cunha, A.S.; Lucena, A.** *Algorithms for the Degree Constrained Minimum Spanning Tree Problem*. In: GRACO - 2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics, 2005, Angra dos Reis. Electronic Notes in Discrete Mathematics, 2005. v. 19. p. 403-409.

[16] **Cunha, A.S.; Lucena, A.** *A hybrid Branch-and-cut Relax-and-cut algorithm for the Degree-constrained Minimum Spanning Tree Problem*. In: International Symposium on Mathematical Programming, 2006, Rio de Janeiro. Proceedings of the 19th International Symposium on Mathematical Programming, 2006.

[17] **Cunha, A.S.; Lucena, A.** *Lower and upper bounds for the degree-constrained minimum spanning tree problem*. Networks, v. 50, p. 55–66, 2007.

[18] **Dantzig, G. B.; Fulkerson, D. R.; Johnson, S. M**. *Solution of a large scale traveling salesman problem*. Operations Research, v. 2, p. 393-410, 1954.

[19] **Desrochers, M. e Laporte, G**. *Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints*. Operations Research Letters, v. 10, p. 27-36, 1991.

[20] **Downey, R.G; Fellows, M. R**. *Fixed-parameter tractability and completeness I: Basic results.* SIAM Journal on Computing, v. 24, p. 873-921, 1995.

[21] **Downey, R.G. e Fellows, M. R**. *Parameterized Complexity*. Springer-Verlag, 1998.

[22] **Edmonds, J.** *Maximum matching and a polyhedron with 0-1 vertices*. J. Res. Nat. Bur. Standards, 69B: p. 125-130, 1965.

[23] **Esau, L.R.; Williams, K.C**. *On teleprocessing network design: Part II. A method for approximating the optimal network*. IBM Systems Journal, v. 5(3), p. 142–147, 1966.

[24] **Freitas, A. T**. *Árvore de subgradiente com pré-fase VNS-Lagrangeana para a árvore geradora com restrição de grau máximo nos vértices*. Dissertação de mestrado, Universidade Federal do Ceará, Fortaleza, CE, Brasil, 2011.

[25] **Fürer, M.; Raghavachari, B..** *Approximating the minimum-degree Steiner tree to within one of optimal*, Journal of Algorithms, v.17(3), p. 409–423, 1994.

[26] **Garey, M. R.; Johnson, D. S**. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, ISBN 0-7167-1045-5. A2.1: ND1, p. 206, 1979.

[27] **Gouveia, L**. *Using the Miller-Tucker-Zemlin constraints to formulate a minimal spanning tree problem with hop constraints*. Computers & Operations Research, v. 22(9), p. 959–970, 1995.

[28] **Gouveia, L**. *Multicommodity Flow Models for Spanning Trees with Hop Constraints*. European Journal of Operational Research, v. 95, p. 178–190, 1996.

[29] **Gouveia, L.; Telhada, J**. *The Multi-weighted Steiner Tree Problem: A Reformulation by Intersection*. Computers and Operations Research, v. 35, p. 3599-3611, 2008.

[30] **Gouveia, L.; Telhada, J**. *Reformulation by Intersection Method on the MST Problem with Lower Bound on the Number of Leaves*. Lecture Notes in Computer Science, v. 6701, p. 83-91, 2011.

[31] **Held, M.; Karp, R. M.** *The traveling salesman problema and minimum spanning trees*. Mathematical Programming, v. 18, p. 1138-1162, 1970.

[32] **Held, M.; Karp, R. M.** *The traveling salesman problema and minimum spanning trees: part II*. Mathematical Programming, v. 1, p. 6-25, 1970.

[33] **Karnaugh, M**. *A New Class of Algorithms for Multipoint Network Optimization*. IEEE Transactions on Communications, v. 24, p. 500–505, 1976.

[34] **Kruskal J**. *On the shortest spanning subtree of a graph and the travelling salesman problem.* Proceedings of the American Mathematical Society, v. 7, p. 48–50, 1956.

[35] **Lemarechal, C**. *An algorithm for minimizing convex functions*. Em Proceedings of IFIP'74 Congress, North Holland, p. 552–556, 1974.

[36] **Magnanti, T. L.; Wolsey, L. A.**. *Optimal Trees,* Handbooks in Operations Research and Management Science - Elsevier, v. 7, c. 9, p. 503-616, 1995.

[37] **Martin, R. K**. *Using separation algorithms to generate mixed integer model reformulations*. Operations Research Letters, v. 10, p. 119–128, 1991.

[38] **Martins, P.; Souza, M. C.** *VNS and second order heuristics for the min-degree constrained minimum sp anning tree problem*. Computers and Operations Research, v. 36, p. 2669–2982, 2009.

[39] **Martinez, L. C**. *Formulações e Algoritmos Sequenciais e Paralelos para o Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo*. Dissertação de Mestrado. Universidade Federal de Minas Gerais, 2012.

[40] **Martinez, L. C.; Cunha, A.** *The min-degree constrained minimum spanning tree problem: Formulations and Branch-and-cut algorithm*. Discrete Applied Mathematics, v. 164, p. 210–224, 2014.

[41] **Martinez, L.C.; Cunha, A**. *A Parallel Lagrangian Relaxation Algorithm for the Min-Degree Minimum Spanning Tree Problem*. In: 2nd International Symposium on Combinatorial Optimization, ISCO 2012, Atenas. Lecture Notes in Computer Science - Combinatorial Optimization. Heildelberg: Springer-Verlag, v. 7422. p. 237-248, 2012.

[42] **Miller, C; Tucker, A; Zemlin, R.** *Integer programming formulation of traveling salesman problems.* Journal of ACM, v. 7, p. 326-9, 1960.

[43] **Myung, Y.; Lee, C.; Tcha, D**. *On the generalized minimum spanning tree problem*. Networks, v. 26(4), p. 231–241, 1995.

[44] **Narula, S. C.; Ho, C. A.**; *Degree-constrained minimum spanning tree.* Computers and Operations Research, v. 7, p. 239-249, 1980.

[45] **Padberg, M.; Wolsey, L**. *Trees and cuts*. Annals of Discrete Mathematics, v. 17, p. 511–517, 1983.

[46] **Ribeiro, C. C; Souza, M. C**. *Variable neighborhood search for the degreeconstrained minimum spanning tree problem*. Discrete Appl. Math. v. 118(1-2), p. 43–54, 2002.

[47] **Rockafellar, R. T.**. Convex Analysis, Princeton Landmarks in Mathematics, 1996.

[48] **Salama, H.F., Reeves, D.S., Viniotis, Y**. The Delay-Constrained Minimum Spanning Tree Problem. In Proceedings of the 2nd IEEE Symposium on Computers and Communications (ISCC '97). IEEE Computer Society, Washington, DC, USA, p. 699–704, 1997.

[49] **Souza, M. C.; Martins, Pedro**. *Skewed VNS enclosing second order algorithm for the degree constrained minimum spanning tree problem*. European Journal of Operational Research, v. 191, p. 677-690, 2008.

[50] **Van Roy, T. J.**. *Cross decomposition for mixed integer programming*. Math. Prog. v. 25, p. 46-63, 1983.

[51] **Wolsey, L. A.** *Integer Programming*, John Wiley & Sons, 1998.