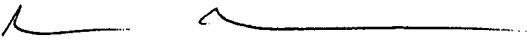


OTIMIZAÇÃO COMBINATÓRIA:
PROBLEMAS DE ÁRVORES EM GRAFOS

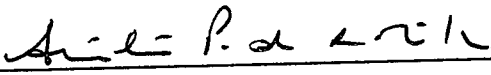
Luidi Gelabert Simonetti

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS
PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE
DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E
COMPUTAÇÃO.

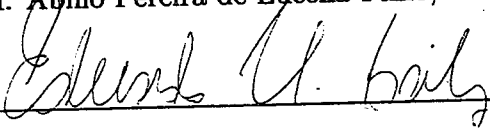
Aprovada por:



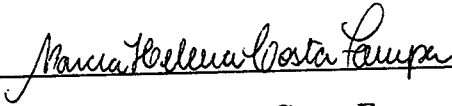
Prof. Nelson Maculan Filho, D.Sc.




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



Prof. Eduardo Uchoa Barboza, D.Sc.



Prof. Marcia Helena Costa Fampa, Ph.D.



Prof. Cid Carvalho de Souza, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

FEVEREIRO DE 2008

SIMONETTI, LUIDI GELABERT

Otimização combinatória: problemas de árvores em grafos [Rio de Janeiro] 2008

X, 86 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2008)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Árvores geradoras 2. Otimização Combinatória 3. Algoritmos Branch-and-Cut

I. COPPE/UFRJ II. Título (série)

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

OTIMIZAÇÃO COMBINATÓRIA: PROBLEMAS DE ÁRVORES EM GRAFOS

Luidi Gelabert Simonetti

Fevereiro/2008

Orientador: Nelson Maculan Filho

Programa: Engenharia de Sistemas e Computação

Neste trabalho, apresentamos modelos e algoritmos para problemas relacionados ao problema de árvore geradora. Três problemas foram estudados: o Problema de Árvore Geradora Mínima com restrição de Nível (PAGMN), o Problema de Árvore Geradora Mínima com restrição de Diâmetro (PAGMD) e o Problema de Árvore Geradora com número Máximo de Folhas (PAGMF). Apresentamos modelos para o PAGMN e o PAGMD, onde o problema é reformulado em grafos direcionados em níveis. Na verdade, isto é equivalente à transformação desses problemas em um Problema de Árvore de Steiner Direcionada (PASD) no grafo em níveis. Os testes computacionais realizados mostraram um grande ganho em relação aos métodos anteriores para os dois problemas. Para o PAGMF, apresentamos uma reformulação direcionada de uma formulação proposta na literatura, que se mostrou mais forte que a original. Introduzimos também uma reformulação do problema em termos do PASD. Para este problema, os teste computacionais também mostraram um ganho em relação aos métodos propostos na literatura.

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

COMBINATORIAL OTIMIZATION: TREE PROBLEMS IN GRAPHS

Luidi Gelabert Simonetti

February/2008

Advisor: Nelson Maculan Filho

Department: Systems Engineering and Computing

In this Thesis, we presented models and algorithms for problems related with the Spanning Tree problem. Three problems were studied: the Hop constrained Minimal Spanning Tree Problem (HMSTP), the Diameter constrained Minimal Spanning Tree Problem (DMSTP) and the Maximum Leaf Spanning Tree Problem (MLSTP). We propose two new formulations for the HMSTP and the DMSTP problems based on layered directed graph. In fact, this is equivalent to transform the problems into a suitable directed Steiner Tree Problem (STP) over that layered graph. Experiments show a significant increase with respect to previous methods. We propose a reformulation for the MLSTP, stronger than the original one, and another transformation to STP. Extensive computational experiments were carried out in order to evaluate the methods. A significant gain in performance with respect to previous methods shows the strength of the approaches.

Sumário

1	Introdução	1
1.1	Árvores Geradoras	1
1.2	Problema de Steiner em Grafos	3
1.3	Métodos de resolução	3
1.3.1	Algoritmos <i>Branch-and-Bound</i>	3
1.3.2	Algoritmos <i>Branch-and-Cut</i>	5
1.4	Contribuições da tese	6
1.5	Organização da tese	7
2	Árvore geradora mínima com restrição de nível	8
2.1	Introdução	8
2.2	Formulação num grafo em níveis	10
2.2.1	Formulação por cortes orientados	12
2.2.2	Formulação multifluxos	14
2.2.3	Formulação multifluxos revisada	15
2.3	Acelerando a resolução do modelo por cortes orientados	17
2.3.1	Heurística Dual	17
2.3.2	Heurística primal	20
2.3.3	Heurística e Buscas Locais para o PSG	21
2.3.4	Heurística e Busca Local para o HMSTP	22

2.3.5	Fixação por custos reduzidos	23
2.3.6	Procedimento inicial	24
2.3.7	Heurísticas e fixação durante o <i>Branch-and-Cut</i>	25
2.4	Resultados	26
2.5	Comentários Finais	33
3	Árvore geradora mínima com restrição de diâmetro	35
3.1	Introdução	35
3.2	Formulação num grafo em níveis	37
3.2.1	Formulação por cortes orientados	41
3.3	Acelerando a resolução do modelo por cortes orientados	42
3.3.1	Heurística e Busca Local para o DMSTP	43
3.4	Resultados	44
3.5	Comentários Finais	52
4	Árvore geradora com número máximo de folhas	53
4.1	Introdução	53
4.2	Uma formulação da literatura	55
4.2.1	Um algoritmo exato	56
4.2.2	Detalhes da implementação	58
4.3	Formulação Direcionada	60
4.4	Formulação Multifluxos	62
4.5	Transformação para o PSG	64
4.5.1	Formulação por cortes orientados	66
4.6	Acelerando a resolução dos modelos	67
4.6.1	Modelo por cortes orientados	67
4.6.2	Heurística primal para o MLSTP - Fújie	68
4.6.3	Heurística primal para o MLSTP - Cobertura	69

4.6.4	Pós-processamento e busca local	70
4.6.5	Teste de pré-processamento	72
4.7	Resultados	73
4.8	Comentários Finais	78
5	Conclusões	79

Lista de Figuras

2.1	Grafo Original	11
2.2	Grafo em nível ($H = 3$)	11
2.3	Exemplo da transformação do grafo original no grafo em nível	11
2.4	Solução original	12
2.5	Solução correspondente	12
2.6	Exemplo da transformação do HMSTP para o PSG ($H = 3$) .	12
3.1	Grafo Original	38
3.2	Grafo em nível ($D = 4$)	38
3.3	Exemplo da transformação do DMSTP para o PSG para D par	38
3.4	Exemplo da transformação do DMSTP para o PSG para D ímpar ($D = 5$)	40
3.5	Detalhe das ligações entre os vértices no retângulo na figura 3.4	40
4.1	Grafo Original	65
4.2	Grafo em nível	65
4.3	Exemplo da transformação do MLSTP para o PSG	65

Lista de Tabelas

2.1	Tamanho das instâncias do HMSTP após a redução	27
2.2	Resultados das formulações do HMSTP para o grupo TC	29
2.3	Resultados das formulações do HMSTP para o grupo TE	29
2.4	Resultados das formulações do HMSTP para o grupo TR	30
2.5	Resultados das formulações do HMSTP para as instâncias maiores	30
2.6	Resultados das heurísticas do HMSTP para o grupo TC	31
2.7	Resultados das heurísticas do HMSTP para o grupo TR	32
2.8	Resultados das heurísticas do HMSTP para o grupo TE	32
2.9	Comparando com a literatura do HMSTP	33
3.1	Resultados do modelo para o DMSTP para o grupo TC	46
3.2	Resultados do modelo para o DMSTP para o grupo TE	46
3.3	Resultados do modelo para o DMSTP para o grupo TR	47
3.4	Resultados das heurísticas do DMSTP para o grupo TC	48
3.5	Resultados das heurísticas do DMSTP para o grupo TE	49
3.6	Resultados das heurísticas do DMSTP para o grupo TR	50
3.7	Resultados dos modelos para o DMSTP para as instâncias criadas por Gouveia	50
3.8	Resultados dos modelos para o DMSTP para as instâncias criadas por Santos	51

4.1	Resultados dos algoritmos para o MLSTP	75
4.2	Resultados das heurísticas para o MLSTP	77

Capítulo 1

Introdução

A evolução da informática e dos métodos de otimização tornou possível a resolução, com garantia de otimalidade, de instâncias de problemas de otimização combinatória de dimensões inimagináveis a a cerca de duas décadas atrás. Com o objetivo de contribuir para essa contínua evolução, escolhemos alguns problemas de otimização combinatória em grafos para testar novos algoritmos de solução para os mesmos. Os problemas aqui escolhidos são algumas variantes do problema de árvore geradora. Essas variantes são o problema de árvore geradora mínima com restrição de nível, *Hop-constrained Minimum Spanning Tree Problem* (HMSTP), o problema de árvore geradora mínima com restrição de diâmetro, *Diameter-constrained Minimum Spanning Tree Problem* (DMSTP), e o problema de árvore geradora com número máximo de folhas, *Maximum Leaf Spanning Tree Problem* (MLSTP).

1.1 Árvores Geradoras

A variante mais famosa da família de problemas de árvore geradora é o Problema de Árvore Geradora Mínima (PAGM). Dado um grafo $G = (V, E)$

com um conjunto de vértices $V = \{1, \dots, n\}$, um conjunto de arestas $e = (i, j) \in E$ com custos associados c_e , desejamos encontrar uma árvore geradora T , onde $T \subset E$ é uma componente conexa e sem ciclos, que contém todos os vértices de V (i.e. existe um único caminho nessa componente entre qualquer par de vértices de V), com custo mínimo. Esse problema pode ser resolvido facilmente, ver em [1]. Entretanto, as variantes aqui estudadas não apresentam essa característica. Enquanto, atualmente, os algoritmos para o PAGM podem resolver instâncias de milhares de vértices, os algoritmos existentes para as variantes aqui estudadas não conseguem resolver, com garantia de otimalidade, instâncias de cem vértices. Com os algoritmos aqui propostos, conseguimos aumentar substancialmente o tamanho das instâncias desses problemas resolvidas com garantia de otimalidade.

O primeiro problema estudado aqui (HMSTP) pode ser definido como encontrar uma árvore geradora mínima onde o caminho único entre um dado vértice (raiz) e qualquer outro vértice dessa árvore não contenha mais do que um dado número de arestas. O segundo (DMSTP) é uma generalização do primeiro, onde desejamos construir uma árvore geradora mínima em que o caminho único entre qualquer par de vértices não contenha mais do que um dado número de arestas. O terceiro e último problema aqui tratado, difere bastante dos outros dois. Embora, nesse problema também se deseje encontrar uma árvore geradora, sua função objetivo é totalmente diferente daquela associado ao PAGM. Nessa variante do problema não lidamos com custos de arestas a minimizar. Nosso objetivo é maximizar o número de folhas da árvore geradora da solução, onde folhas são vértices adjacentes a um único vértice naquela árvore.

1.2 Problema de Steiner em Grafos

Mesmo não sendo objeto de estudo desta tese, a compreensão do Problema de Steiner em Grafos (PSG) e de alguns algoritmos para a sua solução são fundamentais para o entendimento de algumas das formulações e algoritmos de solução aqui propostos.

O PSG pode ser definido como: dado um grafo $G = (V, E)$, um conjunto de vértices terminais $R \subseteq V$ e custos associados às arestas, encontrar uma árvore de G conectando todos os vértices terminais a custo mínimo. O PSG foi um dos primeiros problemas provados NP-difíceis (ver [2]). Mais informações sobre o PSG podem ser encontradas em [3, 4, 5, 6, 7].

1.3 Métodos de resolução

Serão descritos os métodos e técnicas de otimização inteira utilizados na tese. Sem perda de generalidade, fazemos a apresentação, assumindo que os problemas a serem aqui resolvidos encontram-se na forma de minimização. Mais informações sobre os métodos podem ser obtidas em [8].

1.3.1 Algoritmos *Branch-and-Bound*

Genericamente, os algoritmos *Branch-and-Bound* consistem em um procedimento inteligente de enumeração de todas as possíveis soluções de um problema de Otimização Inteira ou Combinatória. Ao invés de enumerar explicitamente todas as possíveis soluções do problema, utiliza-se a combinação de duas estratégias principais: a primeira consiste em particionar o domínio do problema em uma série de subespaços disjuntos, e a segunda consiste em encontrar limites para os problemas definidos em cada subespaço. À pri-

meira estratégia dá-se o nome de *Branching* (ou ramificação) e à segunda, *Bounding* (ou poda).

Seguindo motivações puramente didáticas, vamos considerar a partir de agora, nesta seção, que o Problema de Otimização Linear Inteiro (PI) a ser resolvido tem suas variáveis binárias 0 – 1. Isto não implica em perda de generalidade do método. Vamos considerar que o PI a ser resolvido é descrito como $z = \min\{c^t x : Ax \leq b, x \in \mathbb{B}^n\}$, onde $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, e é denotado por PI_0 .

Suponha que dispomos de um limite inferior z_0 e de um limite superior \bar{z}_0 válidos para PI_0 , tais que $z_0 < \bar{z}_0$. Recorremos, então, a uma forma de enumeração implícita das soluções do mesmo. Esta enumeração é feita da seguinte forma: ao invés de tentar resolver PI_0 , vamos resolver, por exemplo, dois novos problemas PI_1 e PI_2 , definidos respectivamente em espaços idealmente disjuntos. Suponha que, por um critério qualquer (ramificação), tenhamos escolhido uma variável x_j para gerar o particionamento. Então, resolver PI_0 equivale a resolver, em separado, os problemas PI_1 e PI_2 , que se originam de PI_0 , definidos respectivamente por: $\min\{c^t x : Ax \leq b, x_j \leq 0, x \in \mathbb{B}^n\}$ e $\min\{c^t x : Ax \leq b, x_j \geq 1, x \in \mathbb{B}^n\}$, e escolher, entre as duas soluções obtidas, aquela de menor custo.

Observe que em função do particionamento de PI_0 em dois subproblemas PI_1 e PI_2 , desenvolve-se uma árvore de enumeração binária. Cada nó i da árvore corresponde a um subproblema PI_i . A cada um deles, limites superior e inferior podem ser associados.

Considere agora que na investigação do problema PI_1 tenhamos obtido, por um método qualquer, um limite inferior z_1 para PI_1 tal que $z_1 > \bar{z}_0$. Naturalmente, nesse caso, não é necessário prosseguir com o particionamento de PI_1 , já que $z_1 > \bar{z}_0$ comprova não existir na região de viabilidade associada

a PI_1 uma solução ótima para PI_0 . Note, ainda, que no caso em que $z_1 = \bar{z}_0$, poderíamos também excluir de nossa análise o espaço de soluções associado a PI_1 . Isso se aplica, pois o valor de qualquer solução viável em PI_1 teria um valor igual ou superior a \bar{z}_0 .

A nós restaria agora prosseguir na investigação do nó da árvore de enumeração que ainda não foi investigado, correspondente ao problema PI_2 . Suponha que, ao investigá-lo, tenhamos obtido novos limites z_2 e \bar{z}_2 , tais que $z_0 \leq z_2 < \bar{z}_2 \leq \bar{z}_0$. Estes limites não permitem cessar a investigação do problema PI_0 nesse ramo da árvore, de forma que devemos particionar PI_2 , por exemplo, em dois novos subproblemas disjuntos, tendo agora z_2 e \bar{z}_2 como limites globais para PI_0 . O procedimento de ramificação e poda prossegue até que, para todos os nós da árvore de enumeração, tenhamos a garantia de que um limite inferior obtido naquele nó supera ou iguala o melhor limite superior conhecido para o problema, ou que não é possível gerar uma solução viável a partir daquele nó (poda por inviabilidade).

É crucial destacar a generalidade do algoritmo *Branch-and-Bound*. Note que o método não se restringe ao critério de ramificação (em *variáveis*) descrito aqui. Note, também, que em momento algum particularizamos o modo como os limites inferiores e superiores são obtidos. Dependendo do modo como são gerados estes limites, famílias distintas de algoritmos são obtidas. Uma destas famílias é tratada a seguir.

1.3.2 Algoritmos *Branch-and-Cut*

Os algoritmos *Branch-and-Cut* são algoritmos de Planos de Corte inseridos em um esquema de enumeração implícita *Branch-and-Bound*. Nestes algoritmos, os limites inferiores z_i em cada nó da árvore de enumeração são obtidos, inicialmente, a partir da Relaxação Linear da formulação inicial do

subproblema PI_i . Se \bar{x} é uma solução ótima, fracionária, para PI_i , antes de prosseguir com a estratégia de ramificação, tentamos fortalecer os limites inferiores no nó i , através da identificação de desigualdades válidas violadas por \bar{x} . Estas desigualdades, então, incorporadas à formulação do problema PI_i , e um novo limite inferior, reforçado, é obtido para o problema, através da Relaxação Linear correspondente.

Conforme destacado em [8], embora a diferença entre um algoritmo *Branch-and-Cut* e um algoritmo *Branch-and-Bound* pareça pequena, ela implica em uma nova filosofia. Nos algoritmos *Branch-and-Cut*, tipicamente, investe-se mais tempo em cada nó da árvore de enumeração, na tentativa de diminuir o tempo total do algoritmo.

1.4 Contribuições da tese

Nos dois primeiros problemas tratados nesta tese (HMSTP e DMSTP), apresentamos uma transformação desses problemas no PSG. Para cada transformação, foi necessário criar um grafo em níveis G_N , de acordo com o grafo original G . No total, foram necessários três grafos em níveis diferentes. Depois da transformação, apresentamos formulações baseadas nas formulações existentes para o PSG. Também apresentamos algoritmos para a resolução de cada um dos problemas.

Esses dois algoritmos apresentam o que há de melhor na literatura, atualmente, sobre PSG direcionado. Usamos heurísticas primais e dual, buscas locais, fixação por custo reduzido e algoritmo de planos de cortes, todos desenvolvidos para o PSG. Entretanto, todos foram modificados em pequenos detalhes, para aproveitar as características específicas de cada problema. Além disso, propusemos heurísticas primais e buscas locais para cada um dos

problemas.

Para o terceiro problema (MLSTP), propusemos duas novas formulações. Ambas são versões mais fortes de uma formulação existente, obtidas através de uma reformulação direcionada de tal formulação. Uma terceira formulação é proposta, fazendo uma nova transformação para o PSG. Além dos três novos modelos, propusemos um novo teste de pré-processamento, uma heurística primal e uma busca local.

1.5 Organização da tese

Organizamos esta tese pelos três problemas estudados. No capítulo 2, temos a descrição formal do HMSTP e os três modelos propostos para resolvê-lo. Também é apresentada a transformação do problema em que os modelos são baseados. Em seguida, apresentamos o algoritmo *Branch-and-Cut* usado para resolver o problema e as heurísticas primais e dual usadas para acelerar o algoritmo.

No capítulo 3, apresentamos o DMSTP e a adaptação do método usado para resolver o HMSTP para este problema, incluindo a heurística primal específica para o DMSTP.

No capítulo 4, apresentamos o MLSTP e os três modelos propostos. Também apresentamos a transformação do problema para o PSG e adaptamos o método usado no HMSTP para o terceiro modelo. Em seguida, apresentamos as heurísticas primais e buscas locais utilizadas para acelerar os métodos propostos. Por fim, é apresentado um novo teste de pré-processamento para o problema.

Concluimos a tese no capítulo 5, revisando os principais resultados obtidos.

Capítulo 2

Árvore geradora mínima com restrição de nível

2.1 Introdução

O problema de árvore geradora mínima com restrição de nível, *Hop-constrained Minimum Spanning Tree Problem* (HMSTP), é definido como segue: dado um grafo não-direcionado $G = (V, E)$ com um conjunto de vértices $V = \{0, 1, \dots, n\}$, um conjunto de arestas $(i, j) \in E$ com custos associados c_{ij} , $(i, j) \in E$ e um número inteiro positivo $H \leq |V| - 1$, desejamos encontrar uma árvore geradora com custo mínimo, onde o caminho único entre o vértice raiz, vértice 0, e qualquer outro vértice não tenha mais de H níveis (arestas).

O HMSTP é NP-difícil. Este resultado deve-se ao fato de que o problema contém um caso particular, quando $H = 2$, de uma versão NP-difícil do problema de localização de facilidade simples sem capacidade (ver [9, 10, 11]). Em [11] foi mostrado que o HMSTP não pertence a APX, isto é, não pertence à classe de problemas para a qual é possível ter uma heurística com tempo

polinomial, com garantia de um limite aproximado.

A principal aplicação do HMSTP é a modelagem do design de uma rede de telecomunicação centralizada com restrições de qualidade de serviço. O vértice raiz representa a servidora de processador central e os vértices restantes representam os terminais a ligar à servidora. A restrição de nível limita o número de linhas de transmissões (arestas) entre o vértice raiz e qualquer outro e garante certo nível de serviço com respeito a alguma restrição de performance, como disponibilidade e confiabilidade (ver [12]). Disponibilidade é a probabilidade de que todas as linhas de transmissão no caminho entre a servidora (vértice raiz) e o terminal (outro vértice) estejam funcionando. Confiabilidade é a probabilidade de que uma comunicação não seja interrompida por uma falha na linha. Em geral, essas probabilidades diminuem com o número de linhas no caminho. Sendo assim, um caminho com menor número de arestas (níveis) tem uma performance melhor com respeito a disponibilidade e confiabilidade.

Esquemas de limites inferiores para o HMSTP baseados em fluxo de redes foram sugeridos em [13, 14, 15]. Recentemente, DAHL *et al.* [16] propuseram uma formulação envolvendo uma variável associada a cada aresta (em todas as outras formulações é usada uma variável associada a cada aresta e nível) e um número exponencial de restrições, e propuseram um limite inferior, baseado em relaxação lagrangiana. Um recente trabalho de DAHL *et al.* [17] sumariza esses métodos, incluindo a formulação de caminho mínimo em um grafo em níveis, apresentada em [14], que possui os melhores resultados na literatura e as menores diferenças entre relaxação linear e solução ótima (*gaps*).

2.2 Formulação num grafo em níveis

As formulações propostas aqui são oriundas da transformação do HMSTP em PSG. Para transformar o problema, devemos primeiro definir como construir o grafo em níveis a partir do grafo G . Entretanto, para a construção do grafo em níveis é necessário definir o grafo D . Seja $D = (V, A)$ um grafo direcionado, definido a partir de G da seguinte maneira: para cada aresta $(i, j) \in E$, dois arcos (i, j) e (j, i) são definidos em A . Considere o grafo em níveis $G_N = (V_N, A_N)$, que é um grafo direcionado, onde o conjunto de vértices V_N é definido como

$$V_N = \{0\} \cup \{(i, h) : 1 \leq h \leq H, i \in V \setminus \{0\}\}$$

e o conjunto de arcos é definido como

$$\begin{aligned} A_N = & \{(0, (j, 1)) : (0, j) \in A\} \\ & \cup \{((i, h), (j, h + 1)) : (i, j) \in A, i \neq 0, 1 \leq h \leq H - 1\} \\ & \cup \{((i, h), (i, H)) : i \in V \setminus \{0\}, 1 \leq h \leq H - 1\}. \end{aligned}$$

O vértice (i, h) é associado à utilização do vértice i no nível h no grafo original, isto é, o caminho do vértice 0 até o vértice i contém h arestas. Note que o grafo G_N possui $H + 1$ níveis e a cada nível está associada uma cópia do conjunto de vértices de G , com exceção do primeiro nível, composto somente pelo vértice raiz. Na figura 2.3 é apresentado um exemplo da transformação do grafo original (figura 2.1) para o grafo em níveis (figura 2.2).

Considere a árvore de Steiner direcionada mínima em G_N , onde o vértice raiz é o vértice raiz do grafo original, 0, e os vértices terminais são $R = \{(i, H) : i \in V \setminus \{0\}\}$. Não é difícil ver que a árvore geradora com profundidade menor ou igual a H no grafo original corresponde à árvore de Steiner em G_N , com raiz no vértice 0 e com os vértices terminais R . Note que o

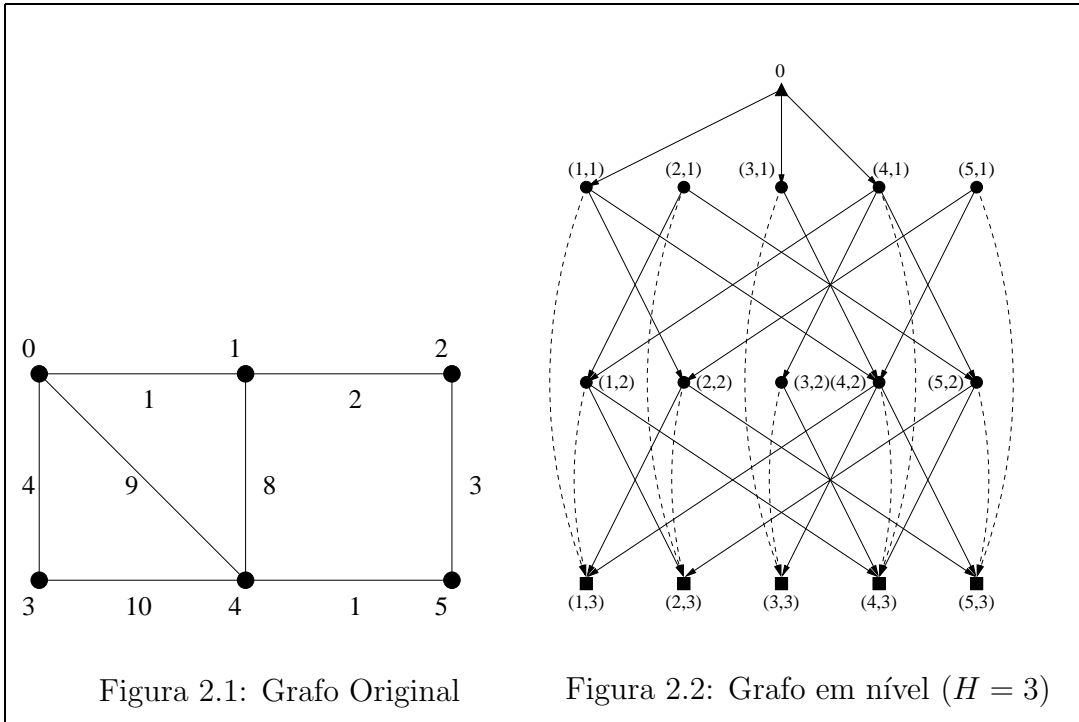


Figura 2.1: Grafo Original

Figura 2.2: Grafo em nível ($H = 3$)

Figura 2.3: Exemplo da transformação do grafo original no grafo em nível

arco $((i, h), (i, H))$ deve ser utilizado sempre que o vértice (i, h) , $i \in V \setminus \{0\}$ e $h \leq H - 1$, está na solução. Na figura 2.6 é apresentado um exemplo da relação entre as soluções no grafo original (figura 2.4) e no grafo em níveis (figura 2.5).

O interessante dessa construção é que associando uma variável binária X_{ij}^h a cada arco $((i, h - 1), (j, h))$ em G_N e associando uma variável binária X_{jj}^h para cada arco $((j, h - 1), (j, H))$ em G_N , podemos usar qualquer modelo para o problema de Steiner no grafo em níveis, para fornecer um modelo válido para o HMSTP. Em seguida, apresentaremos três modelos. O primeiro é uma adaptação para o grafo em níveis da conhecida formulação por cortes orientados, apresentada em [18], para o problema de árvore de Steiner. O segundo modelo é a formulação por multifluxos apresentada em [19], é equivalente ao primeiro, no sentido que os dois modelos possuem o mesmo valor

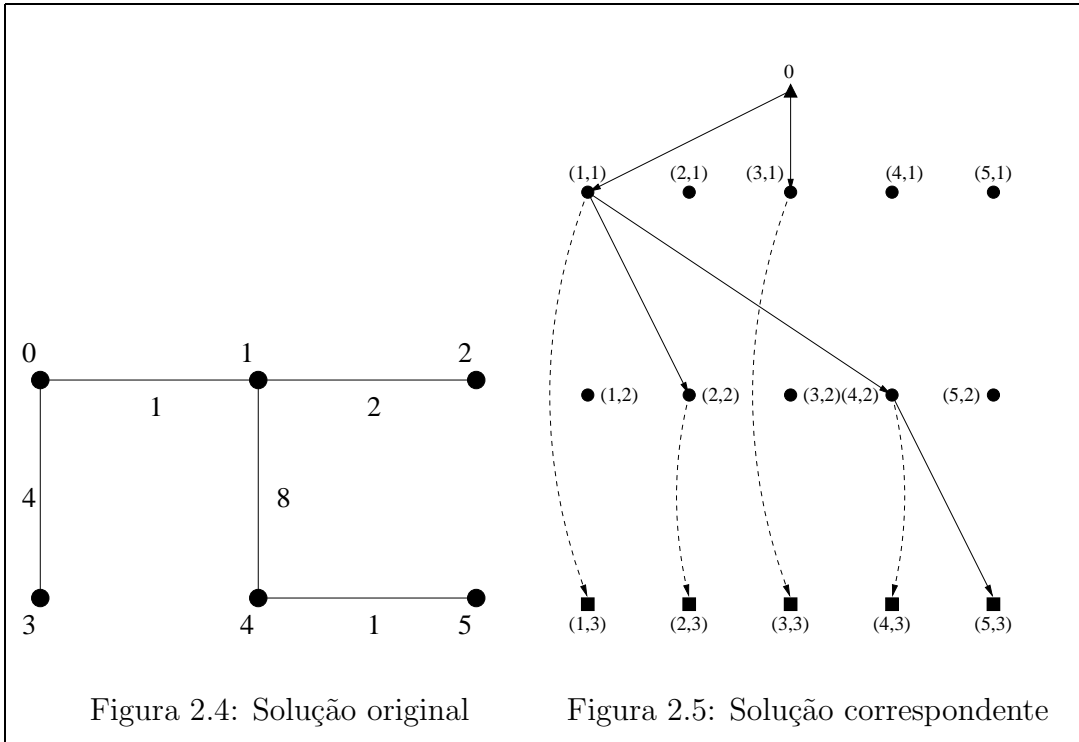


Figura 2.4: Solução original

Figura 2.5: Solução correspondente

Figura 2.6: Exemplo da transformação do HMSTP para o PSG ($H = 3$)

na relaxação linear. Já o terceiro modelo é a versão simplificada do segundo.

2.2.1 Formulação por cortes orientados

Antes de introduzir o modelo, apresentaremos algumas notações usadas para descrever os cortes orientados. Seja S um subconjunto de vértices, onde $0 \notin S$ e $S \cap R \neq \{\emptyset\}$, e \mathcal{S} o conjunto de todos os subconjuntos de vértices S . Cada subconjunto S representa um corte $\delta(S)$ no grafo em níveis, ou seja, o conjunto de arcos $((i, h - 1), (j, h))$, onde $(i, h - 1) \in V_N \setminus S$ e $(j, h) \in S$, e $((i, h), (i, H))$, onde $(i, h) \in V_N \setminus S$ e $(i, H) \in S$. Considerando que $X^h(\delta(S))$ é o somatório das variáveis X_{ij}^h associadas aos arcos do corte $\delta(S)$, podemos

apresentar a formulação a seguir:

$$\min \sum_{(i,j) \in A} c_{ij} \sum_{h=1}^H X_{ij}^h \quad (2.1)$$

$$\text{s.a} \quad \sum_{i \in V \setminus \{0,j\}} X_{ij}^H + \sum_{h=2}^H X_{jj}^h = 1 \quad j \in V \setminus \{0\} \quad (2.2)$$

$$X^h(\delta(S)) \geq 1 \quad S \in \mathcal{S} \quad (2.3)$$

$$X_{0j}^1 \in \{0, 1\} \quad (0, j) \in A \quad (2.4)$$

$$X_{ij}^h \in \{0, 1\} \quad (i, j) \in A, i \neq 0, h = 2, \dots, H \quad (2.5)$$

$$X_{jj}^h \in \{0, 1\} \quad j \in V \setminus \{0\}, h = 2, \dots, H. \quad (2.6)$$

As restrições (2.2) garantem que cada vértice terminal seja visitado uma única vez. As inequações (2.3) são restrições de cortes orientados e garantem que a solução do problema terá pelo menos um arco em $\delta(S)$, garantindo que a solução seja uma árvore de Steiner. A formulação mostrada acima contém um número exponencial de restrições (2.3). Entretanto, podemos facilmente separar, verificar se existe uma restrição violada, em tempo polinomial.

Para separar desigualdades diretamente, temos que achar $|R|$ cortes orientados mínimos entre pares de vértices, respectivamente entre o vértice 0 e cada um dos vértices em R . Um dos melhores algoritmos para esse problema é o *highest-label preflow-push*, que tem complexidade $O(|V|^2 \cdot |A|^{1/2})$, mais informações em [20]. HAO e ORLIN [21] propuseram uma variante desse algoritmo, capaz de encontrar o corte mínimo que separa um dado vértice v de todos os demais vértices no grafo, em um único passo, com a mesma complexidade. É possível adaptar esse algoritmo para encontrar o corte orientado mínimo que separa v de algum outro terminal, incluindo a raiz. Essa adaptação já foi usada em [22, 6]. O algoritmo de separação aqui utilizado é uma adaptação do algoritmo usado por UCHOA [6], que foi adaptado da implementação do algoritmo original de Hao-Orlin disponível na internet em

[23].

2.2.2 Formulação multifluxos

Para comparar a formulação por cortes orientados com outras na literatura, reescrevemos a restrição (2.3) de uma maneira compacta (como feito em [18]), usando fluxo em rede. Para cada $k \in V \setminus \{0\}$, adicionamos as variáveis de fluxo binário y_{ij}^{hk} , indicando quando o arco (i, j) está na posição h no caminho para o vértice k no grafo original (ou alternativamente, quando o arco $((i, h-1), (j, h))$ de G_N está no caminho para o vértice terminal (k, H) na solução da árvore de Steiner), e variáveis y_{jj}^{hj} , indicando quando o vértice j está na posição $h-1$ (ou alternativamente, quando o arco $((j, h-1), (j, H))$ de G_N está na solução). Considere o conjunto de restrições a seguir:

$$\sum_{(0,j) \in E} y_{0j}^{1k} = 1 \quad k \in V \setminus \{0\} \quad (2.7)$$

$$y_{0i}^{1k} - \sum_{j \in V \setminus \{0,i\}} y_{ij}^{2k} = 0 \quad i, k \in V \setminus \{0\}; i \neq k \quad (2.8)$$

$$y_{0k}^{1k} - y_{kk}^{2k} = 0 \quad k \in V \setminus \{0\} \quad (2.9)$$

$$\sum_{j \in V \setminus \{0,i,k\}} y_{ji}^{hk} - \sum_{j \in V \setminus \{0,i\}} y_{ij}^{(h+1)k} = 0 \quad i, k \in V \setminus \{0\}; i \neq k, h = 2, \dots, H-1 \quad (2.10)$$

$$\sum_{j \in V \setminus \{0,k\}} y_{jk}^{hk} - y_{kk}^{(h+1)k} = 0 \quad k \in V \setminus \{0\}, h = 2, \dots, H-1 \quad (2.11)$$

$$\sum_{j \in V \setminus \{0,k\}} y_{jk}^{Hk} + \sum_{h=2}^H y_{kk}^{hk} = 1 \quad k \in V \setminus \{0\} \quad (2.12)$$

$$y_{0j}^{1k} \geq 0 \quad (0, j) \in E, k \in V \setminus \{0\} \quad (2.13)$$

$$y_{ij}^{hk} \geq 0 \quad (i, j) \in E; i, k \in V \setminus \{0\}; h = 2, \dots, H \quad (2.14)$$

$$y_{jj}^{hj} \geq 0 \quad j \in V \setminus \{0\}, h = 2, \dots, H. \quad (2.15)$$

As restrições (2.7) forçam a saída de uma unidade de fluxo do vértice 0 para o vértice k . As restrições (2.8), (2.9), (2.10) e (2.11) garantem a con-

servação de fluxo. Já as restrições (2.12) forçam a chegada de uma unidade de fluxo no vértice k .

Também precisamos de restrições para conectar as variáveis X_{ij}^h com as variáveis y_{ij}^{hk} .

$$y_{0j}^{1k} \leq X_{0j}^1 \quad (0, j) \in E, k \in V \setminus \{0\} \quad (2.16)$$

$$y_{ij}^{hk} \leq X_{ij}^h \quad (i, j) \in E, i \neq 0, k \in V \setminus \{0\}, i \neq k, h = 2, \dots, H \quad (2.17)$$

$$y_{jj}^{hj} \leq X_{jj}^h \quad j \in V \setminus \{0\}, h = 2, \dots, H. \quad (2.18)$$

Substituindo as restrições (2.3) pelas restrições (2.7)-(2.15) e adicionando as restrições (2.16)-(2.18), obtemos uma outra formulação válida para o HMSTP. Essa formulação é uma formulação de fluxo em rede para o problema de Steiner (definida no grafo em níveis), e é sabido que o valor da relaxação linear é igual à relaxação linear do modelo anterior por cortes orientados.

2.2.3 Formulação multifluxos revisada

Podemos reescrever a formulação apresentada na seção 2.2.2, eliminando as variáveis X_{jj}^h e y_{jj}^{hj} , onde $j \in V \setminus \{0\}$ e $h = 2, \dots, H$. Primeiro, podemos observar que as restrições (2.9) e (2.11) nos permite eliminar as variáveis y_{jj}^{hj} . Também podemos observar que na relaxação linear da formulação anterior as restrições (2.16)-(2.18) quando $j = k$ são satisfeitas como igualdade. Então, podemos combinar essas restrições com as restrições (2.9) e (2.11), obtendo

$$X_{0j}^1 = X_{jj}^2 \quad j \in V \setminus \{0\} \quad (2.19)$$

$$\sum_{i \in V \setminus \{0, j\}} X_{ij}^{h-1} = X_{jj}^h \quad j \in V \setminus \{0\}, h = 3, \dots, H, \quad (2.20)$$

permitindo eliminar as variáveis X_{jj}^h . Continuamos com as restrições (2.7), (2.8) e (2.10), com as restrições de conexão (2.16) e (2.17), e modificamos as

restrições (2.2) e (2.12), obtendo a formulação a seguir:

$$\min \sum_{(i,j) \in E} c_{ij} \sum_{h=1}^H X_{ij}^h \quad (2.21)$$

$$\text{s.a.} \sum_{h=2}^H \sum_{i \in V \setminus \{0,j\}} X_{ij}^h + X_{0j}^1 = 1 \quad j \in V \setminus \{0\} \quad (2.22)$$

$$\sum_{(0,j) \in E} y_{0j}^{1k} = 1 \quad k \in V \setminus \{0\} \quad (2.23)$$

$$y_{0i}^{1k} - \sum_{j \in V \setminus \{0,i\}} y_{ij}^{2k} = 0 \quad i, k \in V \setminus \{0\}; i \neq k \quad (2.24)$$

$$\sum_{j \in V \setminus \{0,i,k\}} y_{ji}^{hk} - \sum_{j \in V \setminus \{0,i\}} y_{ij}^{(h+1)k} = 0 \quad i, k \in V \setminus \{0\}; i \neq k, h = 2, \dots, H-1 \quad (2.25)$$

$$\sum_{h=2}^H \sum_{j \in V \setminus \{0,k\}} y_{jk}^{hk} + y_{0k}^{1k} = 1 \quad k \in V \setminus \{0\} \quad (2.26)$$

$$y_{0j}^{1k} \leq X_{0j}^1 \quad (0,j) \in E, k \in V \setminus \{0\} \quad (2.27)$$

$$y_{ij}^{hk} \leq X_{ij}^h \quad (i,j) \in E; i, k \in V \setminus \{0\}; i \neq k, h = 2, \dots, H \quad (2.28)$$

$$y_{0j}^{1k} \geq 0 \quad (0,j) \in E, k \in V \setminus \{0\} \quad (2.29)$$

$$y_{ij}^{hk} \geq 0 \quad (i,j) \in E; i, k \in V \setminus \{0\}; h = 2, \dots, H \quad (2.30)$$

$$X_{0j}^1 \in \{0, 1\} \quad (0,j) \in E \quad (2.31)$$

$$X_{ij}^h \in \{0, 1\} \quad (i,j) \in E, i \neq 0, h = 2, \dots, H. \quad (2.32)$$

As restrições (2.22) forçam que um único arco chegue em j . As restrições (2.23) forçam a saída de uma unidade de fluxo do vértice 0 para o vértice k . As restrições (2.24) e (2.25) garantem a conservação de fluxo. Já as restrições (2.26) forçam a chegada de uma unidade de fluxo no vértice k . As restrições (2.27) e (2.28) são as restrições para conectar as variáveis X_{ij}^h com as variáveis y_{ij}^{hk} .

2.3 Acelerando a resolução do modelo por cortes orientados

Nesta seção, apresentaremos os métodos usados para acelerar a resolução do modelo utilizado.

2.3.1 Heurística Dual

A formulação por cortes orientados apresentada em 2.2.1 já foi bastante estudada para o problema de Steiner em grafos. O problema dual dessa formulação também já foi estudado. WONG [18] propôs uma heurística construtiva, chamada *dual ascent*, para obter soluções duais viáveis de boa qualidade. Melhorias dessa heurística foram propostas em [6]. Essas heurísticas duais apresentam um bom desempenho, demandando baixos tempos computacionais e gerando limites inferiores de qualidade. Além disso, geram um conjunto inicial de cortes, que acelera a resolução da relaxação linear do modelo por cortes orientados.

Para facilitar o entendimento, apresentamos o problema dual da formulação por cortes orientados sem a restrição (2.2):

$$\max \sum_{S \in \mathcal{S}} \pi_S \quad (2.33)$$

$$\text{s.a.} \quad \sum_{S \in \mathcal{S}: a \in \delta(S)} \pi_S \leq c_a \quad a \in A_N \quad (2.34)$$

$$\pi_S \geq 0 \quad S \in \mathcal{S}. \quad (2.35)$$

Associa-se uma variável dual π_S a cada corte direcionado de \mathcal{S} , portanto, o número de variáveis é exponencial. O problema consiste em maximizar a soma das variáveis duais. A restrição (2.34) garante que a soma das variáveis duais associadas aos cortes que contêm um arco a não pode ser superior ao

custo do próprio arco (c_a).

O custo reduzido $\bar{c}_\pi(a)$ da variável associada a um arco a em relação a uma solução dual viável π é definido a seguir:

$$\bar{c}_\pi(a) = c_a - \sum_{S \in \mathcal{S} : a \in \delta(S)} \pi_S.$$

Dizemos que um arco está saturado, se o seu custo reduzido for zero. Seja $G_\pi = (V_N, A'_N)$ o subgrafo de G_N contendo apenas os arcos saturados, \mathcal{R} uma componente fortemente conexa de G_π e $V(\mathcal{R})$ o conjunto de vértices de \mathcal{R} . \mathcal{R} será uma componente-raiz, se obedecer às seguintes condições:

- $V(\mathcal{R})$ contém pelo menos um terminal.
- $V(\mathcal{R})$ não contém a raiz.
- Não existe um caminho em G_π de um terminal r , onde $r \in R \cup \{0\}$ e $r \notin \mathcal{R}$, até \mathcal{R} .

Dado um componente-raiz \mathcal{R} , define-se $w(\mathcal{R})$ como o conjunto de vértices que alcançam \mathcal{R} em G_π , incluindo os próprios vértices de \mathcal{R} , ou seja, $V(\mathcal{R})$. O conjunto de arcos não saturados incidentes em $w(\mathcal{R})$, arcos (i, j) , onde $i \in V_N \setminus w(\mathcal{R})$ e $j \in w(\mathcal{R})$, é representado por $\delta(w(\mathcal{R}))$. A variável dual associada ao corte definido por $w(\mathcal{R})$, ou seja, $\delta(w(\mathcal{R}))$, é denotada por $\pi_{w(\mathcal{R})}$.

O algoritmo 2.1 apresenta o pseudocódigo do algoritmo *dual ascent*, que pode ser aplicado em qualquer solução dual viável π .

Começamos com $\pi = 0$, com isso, só temos os arcos saturados $((i, h), (i, H))$, onde $1 \leq h \leq H - 1$. No nosso problema, cada terminal é uma componente-raiz e só deixará de ser quando existir um caminho em G_π da raiz até ele. As outras duas formas de deixar de ser um componente-raiz, que não ocorrem no nosso problema, seriam existir um caminho em G_π entre terminais

Algoritmo 2.1 Dual_Ascent(π)

- 1: Cria G_π com os arcos saturados em relação a π
 - 2: **enquanto** existe uma componente-raiz \mathcal{R} em G_π **faça**
 - 3: $S \leftarrow w(\mathcal{R})$
 - 4: Aumente π_S até que algum arco em $\delta(S)$ fique saturado
 - 5: Adicione a G_π os novos arcos saturados
 - 6: **fim enquanto**
 - 7: **return** π
-

que não pertençam a um mesmo componente-raiz, e não conter a raiz. Isso ocorre, porque cada componente-raiz não muda durante o algoritmo, já que não existe arco saindo dos terminais no grafo G_N . Como a cada iteração pelo menos um arco é adicionado a G_π , o máximo de iterações é $|A_N|$ e no fim do algoritmo existirá um caminho entre a raiz e cada terminal.

O algoritmo 2.1 exige que em cada iteração seja escolhida uma componente-raiz, mas não especifica exatamente qual deve ser a escolhida. Em [6] é apresentado um estudo sobre o impacto dessa escolha. Aqui utilizamos três tipos de escolha: circular, MinArcos e MinArcos+MinSaturados. A escolha circular, a cada iteração, percorre a lista de componentes-raízes, escolhendo uma componente-raiz seguinte à selecionada na iteração anterior; quando se chega ao final da lista de componentes-raízes, retorna-se ao começo. Na escolha MinArcos, escolhe-se uma componente que induz um corte com o mínimo número de arcos. Já na escolha MinArcos+MinSaturados, escolhe-se uma componente, usando o critério principal de MinArcos e, como critério de desempate, o MinSaturados (na escolha MinSaturados, escolhe-se uma componente que induz um corte que satura o mínimo número de arcos).

Melhorando a solução da heurística dual

O procedimento *Dual Adjustment* é um método heurístico proposto em [6], para tentar aumentar o valor da solução dual encontrada pelo algoritmo *dual ascent*. Esse procedimento consiste em reduzir o valor de algumas variáveis duais para possibilitar o aumento de outras. Esse método usa uma solução primal como guia, para reduzir o valor das variáveis duais e, em seguida, aplica o *dual ascent* na nova solução dual gerada.

2.3.2 Heurística primal

Depois de terminada a execução do *dual ascent*, haverá pelo menos um caminho da raiz até qualquer terminal em G_π . Sendo assim, podemos usar heurísticas nesse grafo para tentar encontrar uma solução primal para o problema. Normalmente, o grafo G_π contém um número muito menor de arcos que o grafo G_N , o que garante um tempo computacional menor para uma heurística, normalmente, diretamente dependente do número de arcos. Além disso, esse grafo, na prática, se mostra um bom subgrafo de G_N para obter uma solução primal de boa qualidade.

Pensando nisso, escolhemos dois conjuntos de procedimentos para tentar encontrar uma boa solução primal. O primeiro conjunto são procedimentos feitos para o PSG. O primeiro procedimento desse conjunto é a heurística proposta por TAKAHASHI e MATSUYAMA [24] para o problema de Steiner. Em seguida, utilizamos duas buscas locais (*Node* [25, 26] e *Key-path* [27]) para tentar melhorar a solução obtida pelo primeiro procedimento. O segundo conjunto é formado por uma heurística construtiva e uma busca local feitos diretamente para o HMSTP. Nesse segundo conjunto, usaremos um outro grafo gerado de G_π .

2.3.3 Heurística e Buscas Locais para o PSG

Apresentaremos de forma resumida a heurística construtiva e as duas buscas locais usadas aqui.

O algoritmo de TAKAHASHI e MATSUYAMA [24] é uma modificação do tradicional algoritmo proposto por PRIM [1], para a resolução do PAGM. Neste algoritmo, uma árvore geradora de custo mínimo é construída, seqüencialmente, a partir de um nó raiz. Em cada passo, um vértice é adicionado à árvore geradora em construção, através de uma aresta de menor custo. Após $|V| - 1$ passos, todos os vértices do grafo terão sido incluídos na árvore, que se torna, assim, geradora. A heurística correspondente para o PSG opera de forma análoga. Em cada iteração, adicionamos à solução o vértice terminal mais próximo, ou seja, aquele que se encontra a uma distância mínima da árvore de Steiner em construção. No processo, adicionamos à solução todos os vértices no caminho mínimo que leva ao vértice terminal escolhido. Dessa maneira, em $|R|$ iterações, uma árvore de Steiner é gerada para o conjunto de vértices terminais R e raiz em 0.

A busca local *Node* [25, 26] consiste em procurar vizinhos de uma solução, adicionando ou removendo um vértice de Steiner numa solução. Lembramos que vértice de Steiner são todos os vértices de uma árvore de Steiner que não é a raiz ou um vértice terminal.

A busca local *Key-path* [27] consiste em procurar vizinhos de uma solução encontrada, desconectando um terminal e reconectando com o menor caminho até o resto da solução. Se encontrar um caminho menor do que o anterior, uma nova solução com custo menor foi encontrada. Esse procedimento continua, até não conseguir reconectar nenhum outro vértice terminal com custo menor.

2.3.4 Heurística e Busca Local para o HMSTP

Além das heurísticas para o PSG, também utilizamos uma heurística específica para o HMSTP. Essa heurística, que chamaremos de Hop-Prim, pode ser separada em duas partes. A primeira parte é uma adaptação do algoritmo de PRIM [1], onde criamos uma solução viável para o HMSTP. A cada iteração dessa primeira parte, construímos um nível para a árvore geradora T , até conectarmos todos os vértices do grafo. Essa primeira parte tem no máximo H iterações; se chegarmos na última iteração e não construímos uma árvore geradora, a instância é inviável. Cada iteração consiste em achar as arestas com menores custos que ligam um vértice i , $i \in I$, onde I é o conjunto de vértices conectados na iteração anterior, a um outro vértice j , $j \notin T$, que ainda não foi adicionado à árvore geradora em construção. Na primeira iteração, o conjunto I só contém a raiz, $I = \{0\}$, e adicionamos as arestas $(0, j) \in E$. Na próxima iteração, o conjunto I será formado pelos vértices j das arestas $(0, j)$.

Depois de construir uma solução viável com a primeira parte da heurística Hop-Prim, usamos um procedimento de busca local para melhorar essa solução na segunda parte da heurística. A busca local consiste em trocar uma aresta (i, j) da solução corrente por uma outra que não pertence à solução (k, j) . Essa aresta (k, j) tem que respeitar alguns critérios: menor custo ($c(k, j) < c(i, j)$) ou mesmo custo ($c(k, j) = c(i, j)$), só que k está num nível menor que i (mais próximo da raiz 0); não formar ciclos; respeitar a restrição de nível tanto para j como para os seus filhos (vértices que possuem j no caminho único do vértice raiz 0 e eles).

A heurística Hop-Prim não é usada no grafo original, com exceção da primeira parte, que pode usar (se necessário) o grafo original. Ela usa o grafo G_π^{Ori} , onde $G_\pi^{\text{Ori}} = (V, A')$ é criado a partir de G_π . Se existe um arco

$(0, (i, 1))$, $i \in V$ no grafo G_π , adicionamos a aresta $(0, i)$ a G_π^{Ori} . Se existe pelo menos um dos arcos $((i, h-1), (j, h))$, $(i, j) \in E$ e $h = 2, \dots, H$, em G_π , adicionamos a aresta (i, j) a G_π^{Ori} . Já os arcos $((i, h), (i, H))$, $i \in V$ e $h = 1, \dots, H-1$ são ignorados. Na verdade estamos transformando o grafo G_π num subgrafo do grafo original G . Esse novo grafo G_π^{Ori} tem propriedades semelhantes a G_π . Possui um número menor de aresta do que G e contém pelo menos uma solução viável para o problema (se existir uma solução viável em G). A primeira parte da heurística Hop-Prim, onde criamos uma solução viável, tenta criar uma solução usando só as arestas de G_π^{Ori} . Entretanto, se numa iteração não existe uma aresta (i, j) , onde $i \in I$ e $j \notin T$, no grafo G_π^{Ori} , mas existe no grafo original G , adicionamos a menor aresta (i, j) de G , onde $i \in I$ e $j \notin T$. Isso é feito para garantir uma solução viável para o HMSTP no final da primeira parte da heurística.

2.3.5 Fixação por custos reduzidos

Depois de calculado um limite inferior, solução dual, e um limite superior, solução primal, podemos tentar eliminar, fixar algumas variáveis do problema, usando o custo reduzido. Para mais informações sobre fixação de variáveis por custo reduzido ver [8].

Seja π uma solução dual viável de valor $v(\pi)$ e seja Z o valor da melhor solução primal conhecida. Pode-se fixar a variável X_{ij}^h , referente ao arco $((i, h-1), (j, h)) \in A_N$, em zero, se a seguinte condição for observada:

$$v(\pi) + \bar{c}_\pi((i, h-1), (j, h)) \geq Z.$$

A fixação em zero baseia-se numa redução ao absurdo. Supõe-se que o arco $((i, h-1), (j, h))$, referente à variável X_{ij}^h , está na solução ótima. Então, pode-se aumentar a variável dual em $\bar{c}_\pi((i, h-1), (j, h))$. O custo da solução

passa de $v(\pi)$ para $v(\pi) + \bar{c}_\pi((i, h - 1), (j, h))$. Se esse valor não for menor que a melhor solução primal conhecida, chega-se a uma contradição.

Esse teste pode ser melhorado para o problema de Steiner, como mostrado em [28, 29], tornando maiores as chances de fixar uma variável. A suposição de que o arco $((i, h - 1), (j, h))$ está na solução ótima tem outras conseqüências. A primeira é que todo arco $((i, h - 1), (j, h))$ na solução deve ter um caminho da raiz até $(i, h - 1)$. A segunda é que deve haver um caminho entre (j, h) e algum terminal, isso se (j, h) não for um terminal. Então, podemos melhorar o teste de fixação, calculando o menor caminho entre a raiz e o vértice $(i, h - 1)$, usando como custo dos arcos o custo reduzido. O mesmo pode ser feito para o vértice (j, h) e um terminal, só que, no nosso caso, a distância sempre será zero, já que todo vértice tem um arco saindo para um terminal com custo zero, $((j, h), (j, H))$, conseqüentemente saturado e com custo reduzido zero. O teste de fixação melhorado é apresentado a seguir:

$$v(\pi) + \bar{c}_\pi((i, h - 1), (j, h)) + Q(0, (i, h - 1)) \geq Z.$$

O novo termo $Q(0, (i, h - 1))$ é a menor distância entre a raiz e o vértice $(i, h - 1)$ no grafo G_N e com os custos dos arcos substituídos pelos custos reduzidos correspondentes. Esse valor é não negativo, portanto, aumentando a probabilidade de eliminar o arco $((i, h - 1), (j, h))$.

2.3.6 Procedimento inicial

No início do nosso algoritmo, geramos limites superiores e inferiores, usando os procedimentos mencionados nesta seção. Na tentativa de melhorar os limites obtidos, decidimos utilizar esses procedimentos da forma explicada a seguir. Utilizamos os três tipos de escolha da componente-raiz para o *dual ascent*: circular, MinArcos e MinArcos+MinSaturados. Então, para cada

vez que executamos o *dual ascent* (uma para cada tipo de escolha), geramos um grafo G_π diferente. Depois, executamos a heurística e busca local para o HMSTP no grafo G_π^{Ori} (transformação do grafo G_π). Se melhorarmos o limite superior (solução viável), fazemos o teste de fixação por custos reduzidos. Em seguida, executamos a heurística e buscas locais para o PSG no grafo G_π . Se melhorarmos o limite superior, fazemos o teste de fixação por custos reduzidos. Com a solução encontrada no *dual ascent* e com a melhor solução primal encontrada, utilizamos o procedimento *dual adjustment* para tentar melhorar o limite inferior. No final do procedimento inicial, teremos pelo menos três limites inferiores e seis limites superiores. Ficamos com o maior dos limites inferiores e o menor dos limites superiores. Em seguida, fazemos o último teste de fixação por custos reduzidos, utilizando esses dois valores.

2.3.7 Heurísticas e fixação durante o *Branch-and-Cut*

O algoritmo *Branch-and-Cut* (BC) demora a convergir (demora a resolver a relaxação linear) em algumas instâncias, mesmo com os procedimentos iniciais, as heurísticas e a fixação de variáveis. Por causa desse comportamento, decidimos incorporar ao BC alguns procedimentos usados no início do algoritmo, na tentativa de torna-lo mais robusto (mais independente do êxito do procedimento inicial).

O primeiro procedimento utilizado foi a fixação de variáveis por custo reduzido. Utilizamos o teste de fixação só no nó zero do BC. Inicialmente, a idéia era rodar o teste em todas as iterações do nó zero do BC, mas depois de alguns testes verificamos que não era necessário. Só fazemos o teste quando um dos três critérios é verdadeiro: (1) se não rodamos o teste nas últimas p_1 iterações do BC; (2) se uma nova solução foi encontrada; (3) se do último

teste rodado até a iteração corrente, o valor da relaxação linear aumentou pelo menos p_2 unidades. Utilizamos os parâmetros 5 e 0.5 para p_1 e p_2 , respectivamente.

O segundo e último procedimento utilizado foi a heurística detalhada em 2.3.3 com as buscas locais. A heurística é usada em todas as iterações do BC; a única diferença entre cada iteração são os custos dos arcos. Os custos dos arcos são modificados usando a informação da relaxação linear. O custo de um arco $((i, h-1), (j, h))$ é multiplicado pelo inverso do seu uso pela relaxação linear, ou seja, $c'((i, h-1), (j, h)) = c(i, j) * (1 - X_{ij}^h)$. Para não tornar o uso desse segundo procedimento muito dispendioso, não utilizamos as buscas locais em todas as iterações, só utilizamos quando a solução encontrada pela heurística na iteração corrente multiplicada por um fator p_3 for menor que a melhor solução encontrada até o momento. Utilizamos o valor de 0.85 para p_3 .

2.4 Resultados

Nesta seção, comparamos a formulação por cortes orientados e multifluxos revisada com a melhor formulação da literatura para o HMSTP.

Usamos instâncias de grafo completo de 21 a 161 vértices para testar os algoritmos. Podemos dividir as instâncias em 3 grupos, dois grupos (TC e TE) têm custo euclidiano, e o terceiro (TR) tem custo randômico. Nos grupos com custo euclidiano, o que muda é a localização do vértice raiz: no grupo TC, a raiz fica no centro do grafo e, no grupo TE, a raiz fica num dos extremos do grafo. Cada instância é resolvida com os parâmetros de nível H igual a 3, 4 e 5.

Para reduzir o tamanho de cada instância, usamos o teste a seguir. Para

mais detalhes, ver [13]. Serão eliminadas as arestas que não atendam à seguinte condição:

$$c_{ij} < c_{0j}, \quad i, j \in V \setminus \{0\}.$$

Se a aresta (i, j) tem custo maior do que a aresta $(0, j)$, nenhuma solução ótima possui a aresta (i, j) . No caso de ter o mesmo valor, podemos substituir (i, j) pela aresta $(0, j)$, que não altera o valor da solução e não compromete a restrição de nível. O teste de redução é aplicado em todas as instâncias antes de aplicar os algoritmos. A tabela 2.1 apresenta o percentual restante de cada instância, após a aplicação do teste.

Grupo	V								
	20	30	40	50	60	80	100	120	160
TC	27%	27%	28%	25%	25%	26%	26%	28%	53%
TR	45%	51%	46%	-	52%	47%	-	-	-
TE	70%	73%	68%	71%	70%	68%	71%	76%	78%

Tabela 2.1: Tamanho das instâncias do HMSTP após a redução

Os resultados computacionais foram obtidos em um PC Intel Core 2 Duo, 2.2 GHz com 2Gb de RAM, utilizando-se o programa XPRESS 2007A para resolver as relaxações lineares e a programação inteira, quando necessário. Comparamos nossos resultados com os resultados obtidos em [17]. Os resultados computacionais dos modelos apresentados em [17] foram obtidos em um PC Pentium IV, 2.4 GHz com 769MB de RAM, utilizando o programa CPLEX 7.1 para resolver as relaxações lineares e a programação inteira. Além dos resultados obtidos em [17], Gouveia disponibilizou outros resultados, para outras instâncias, usando o mesmo algoritmo e computador usado em [17].

Os resultados das duas formulações propostas aqui para as instâncias até

80 vértices são apresentados nas tabelas 2.2, 2.3 e 2.4. Cada tabela é referente aos resultados de um grupo de instâncias. A primeira coluna indica o nome da instância, que é formado por duas partes; as duas letras iniciais indicam a que grupo a instância pertence (TC, TE ou TR); em seguida, o número de vértices da instância, sem contar o vértice raiz. A segunda coluna indica o valor de H. A terceira coluna indica o valor da solução ótima. Nas colunas seguintes, indicamos o valor da relaxação linear e o tempo computacional da melhor formulação apresentada em [17] e de cada formulação proposta aqui (Hopcut e Multifluxo). Essa formulação vai ser chamada aqui por HopMCF. Os resultados da formulação HopMCF apresentados nessas tabelas possuem dois tempos computacionais: o primeiro é o tempo necessário para resolver a relaxação linear, e o segundo, entre parênteses, é o tempo necessário para obter a solução inteira ótima. Quando os dois tempos estiverem com o valor -, significa que não testaram o modelo nessa instância. Já quando o segundo tempo for -, significa que não tentaram obter a solução ótima inteira. Os resultados da formulação por cortes orientados (Hopcut) apresentados nessas tabelas também possuem dois tempos computacionais: o primeiro é o tempo sem as melhorias apresentadas na seção 2.3, e o segundo, entre parênteses, com as melhorias.

Como mencionado anteriormente, as formulações por cortes orientados e a de multifluxo revisada apresentam a mesma relaxação linear; a diferença está no tempo computacional gasto para resolver as instâncias. Constatamos que a relaxação linear das duas formulações em todas as instâncias é o ótimo do problema. Não foi possível resolver todas as instâncias utilizando a relaxação linear da formulação multifluxo revisada, devido ao grande tempo computacional necessário. Pode-se constatar que as melhorias apresentadas na seção 2.3 reduziram consideravelmente o tempo necessário para resolver

Nome	H	Ótimo	HopMCF		Hopcut		Multifluxo	
			LP	T(s)	LP	T(S)	LP	T(S)
TC20	3	340	339	0 (0)	340	0.04 (0)	340	0.17
	4	318	318	0 (0)	318	0.04 (0)	318	0.86
	5	312	312	0 (0)	312	0.08 (0)	312	0.45
TC30	3	506	-	-	506	0.13 (0.04)	506	1.61
	4	448	-	-	448	0.19 (0.03)	448	47.2
	5	426	-	-	426	0.33 (0)	426	288.1
TC40	3	609	604.5	2 (40)	609	0.34 (0.06)	609	7.03
	4	548	547	8 (3)	548	1.25 (0.07)	548	305
	5	522	522	13 (0)	522	4.11 (0.29)	522	2014
TC50	3	746	-	-	746	0.74 (0.18)	746	208
	4	683	-	-	683	4.46 (0.22)	683	53014
	5	646	-	-	646	12.3 (0.02)	-	-
TC60	3	866	861.6	21 (578)	866	2.3 (0.35)	866	27667
	4	781	778.3	120 (2052)	781	10.6 (1.26)	-	-
	5	734	734	159 (1)	734	34.3 (0.56)	-	-
TC80	3	1072	1069	160 (3470)	1072	11.7 (0.49)	-	-
	4	981	976.5	1811 (23659)	981	175 (46.9)	-	-
	5	922	920.6	4198 (7590)	922	429 (6.57)	-	-

Tabela 2.2: Resultados das formulações do HMSTP para o grupo TC

Nome	H	Ótimo	HopMCF		Hopcut		Multifluxo	
			LP	T(s)	LP	T(S)	LP	T(S)
TE20	3	449	443.8	2 (37)	449	0.21 (0.03)	449	1.08
	4	385	384.5	6 (5)	385	0.31 (0.02)	385	5.46
	5	366	364	14 (108)	366	1.19 (0.05)	366	16.1
TE30	3	597	-	-	597	1.53 (0.12)	597	24.2
	4	521	-	-	521	3.52 (0.14)	521	90.6
	5	483	-	-	483	6.65 (0.42)	483	249
TE40	3	708	701.7	175 (1984)	708	3.31 (0.13)	708	3485
	4	627	625.3	969 (9797)	627	10.6 (0.14)	627	20866
	5	590	588.1	2794 (23052)	590	29.7 (0.41)	-	-
TE50	3	1366	-	-	1366	9.79 (0.12)	1366	22949
	4	1212	-	-	1212	54.3 (0.54)	-	-
	5	1110	-	-	1110	100 (2.21)	-	-
TE60	3	1525	1503.4	2548 (31825)	1525	19 (0.48)	-	-
	4	1336	1314.5	23402 (> 11 dias)	1336	115 (3.77)	-	-
	5	1225	1212.5	73300 (-)	1225	255 (0.46)	-	-
TE80	3	1806	1792.5	11367 (-)	1806	76 (1.24)	-	-
	4	1558	1544.5	160127 (-)	1558	549 (4.7)	-	-
	5	1442	-	-	1442	2143 (226)	-	-

Tabela 2.3: Resultados das formulações do HMSTP para o grupo TE

Nome	H	Ótimo	HopMCF		Hopcut		Multifluxo	
			LP	T(s)	LP	T(S)	LP	T(S)
TR20	3	168	168	0 (0)	168	0.01 (0)	168	0.23
	4	146	146	0 (0)	146	0.04 (0)	146	2.07
	5	137	137	0 (0)	137	0.07 (0)	137	7.22
TR30	3	229	-	-	229	0.21 (0.03)	229	14.7
	4	174	-	-	174	0.64 (0.1)	174	230
	5	155	-	-	155	0.95 (0)	155	600
TR40	3	176	176	2 (0)	176	0.25 (0.05)	176	7.84
	4	149	148.3	9 (3)	149	1.16 (0.13)	149	119
	5	139	139	26 (1)	139	2.02 (0)	139	93
TR60	3	213	-	-	213	2.76 (0.17)	213	58761
	4	152	-	-	152	9.59 (0.18)	-	-
	5	124	-	-	124	10.6 (0.02)	-	-
TR80	3	208	208	21 (3)	208	1.36 (0.14)	208	86.2
	4	180	180	676 (4)	180	9.87 (0.36)	-	-
	5	164	164	1271 (6)	164	29.8 (0.92)	-	-

Tabela 2.4: Resultados das formulações do HMSTP para o grupo TR

a formulação por cortes orientados.

Os resultados da formulação por cortes orientados (Hopcut), com as melhorias apresentadas na seção 2.3, para as instâncias maiores que 80 vértices são apresentados na tabela 2.5. A primeira coluna indica o valor de H . Nas colunas seguintes indicamos o nome da instância, o valor da solução ótima, a relaxação linear e o tempo computacional, para as instâncias do grupo TC e TE. Para a instância TE160 e $H = 5$, apresentamos dois tempos computacionais: o segundo, entre parênteses, é o tempo necessário a mais para obter a solução inteira ótima. Esse foi o único caso que não resolvemos só com a relaxação linear da formulação Hopcut, foi necessário resolver 4 nós da árvore de busca do *Branch-and-Cut*.

H	TC				TE			
	Nome	Ótimo	LP	T(s)	Nome	Ótimo	LP	T(S)
3		1259	1259	15.3		2092	2092	9.85
4	TC100	1166	1166	190	TE100	1788	1788	356
5		1104	1104	251		1625	1625	36.6
3		1059	1059	2.68		1267	1267	15.6
4	TC120	926	926	25.9	TE120	1074	1074	608
5		853	853	103		969	969	3992
3		1357	1357	163		1496	1496	76.2
4	TC160	1133	1133	643	TE160	1229	1229	5626
5		1039	1039	11335		1107	1106.5	57011 (6704)

Tabela 2.5: Resultados das formulações do HMSTP para as instâncias maiores

Os resultados das heurísticas estão nas tabelas 2.6, 2.7 e 2.8. A primeira coluna das tabelas indica o nome da instância, a segunda indica o valor de H e a terceira coluna indica o valor da solução ótima. Nas cinco colunas seguintes, indicamos o melhor valor obtido com as heurísticas duais, a melhor solução encontrada por todas as heurísticas primais, a melhor solução encontrada pela heurística Hop-Prim, o número de variáveis fixadas pelo teste de custo reduzido e o tempo computacional total do procedimento inicial (heurísticas duais e primais, e fixação de variáveis). As últimas três colunas são referentes à heurística primal, ao número de variáveis fixadas, ao tempo computacional total das heurísticas e às fixações de variáveis durante o BC.

Nome	H	Ótimo	Procedimento Inicial				Durante BC			
			Dual	Primal	Hop-Prim	Var. Fix.	T(S)	Primal	Var. Fix.	T(S)
TC20	3	340	340	340	340	-	0	-	-	-
	4	318	318	318	318	-	0	-	-	-
	5	312	312	312	312	-	0	-	-	-
TC30	3	506	502	522	524	134	0.01	506	249	0.01
	4	448	448	452	452	640	0	448	4	0
	5	426	426	426	426	-	0	-	-	-
TC40	3	609	601	609	623	720	0.01	-	96	0
	4	548	540	548	552	1105	0	-	115	0
	5	522	516	524	526	1413	0	522	242	0
TC50	3	746	734	773	773	92	0.02	752	859	0.04
	4	683	679	683	689	1684	0.02	-	88	0.01
	5	646	646	646	656	-	0.02	-	-	-
TC60	3	866	857	892	919	218	0.03	866	1407	0.04
	4	781	775	795	801	1199	0.03	785	1132	0.02
	5	734	734	738	748	3306	0.02	734	4	0
TC80	3	1072	1066	1084	1148	1418	0.05	1072	1316	0.01
	4	981	973	995	1017	1601	0.06	981	709	0.06
	5	922	920	934	942	3954	0.05	922	1342	0.08
TC100	3	1259	1237	1290	1350	31	0.08	1271	2585	0.12
	4	1166	1158	1198	1219	318	0.11	1166	3148	0.14
	5	1104	1098	1116	1116	5416	0.07	1104	1362	0.11
TC120	3	1059	1051	1102	1174	65	0.2	1059	4017	0.06
	4	926	921	962	1001	551	0.32	926	355	0.11
	5	853	849	857	900	4985	0.39	853	8649	0.11
TC160	3	1357	1349	1426	1715	151	0.89	1357	3122	0.16
	4	1133	1130	1163	1286	15693	1.38	1133	1242	0.1
	5	1039	1033	1055	1116	32124	1.48	1039	4084	0.18

Tabela 2.6: Resultados das heurísticas do HMSTP para o grupo TC

Na tabela 2.9, compara-se o desempenho da formulação por corte orientado com as melhorias apresentadas na seção 2.3, com a melhor formulação

Nome	H	Ótimo	Procedimento Inicial					Durante BC			
			Dual	Primal	Hop-Prim	Var. Fix.	T(S)	Primal	Var. Fix.	T(S)	
TR20	3	168	168	168	168	-	0	-	-	-	
	4	146	146	146	146	-	0	-	-	-	
	5	137	137	137	137	-	0	-	-	-	
TR30	3	229	224	229	237	880	0.01	-	48	0	
	4	174	168	174	181	1295	0.01	-	56	0	
	5	155	155	155	157	-	0	-	-	-	
TR40	3	176	172	177	177	1447	0.01	176	85	0.01	
	4	149	147	149	151	2238	0	-	48	0	
	5	139	139	139	139	-	0	-	-	-	
TR60	3	213	206	217	217	3223	0.03	213	303	0.01	
	4	152	151	153	155	5457	0.02	152	6	0.01	
	5	124	124	124	124	-	0.02	-	-	-	
TR80	3	208	206	208	209	6090	0.02	-	35	0.01	
	4	180	178	180	182	8989	0.03	-	120	0	
	5	164	162	164	167	11905	0.02	-	182	0	

Tabela 2.7: Resultados das heurísticas do HMSTP para o grupo TR

Nome	H	Ótimo	Procedimento Inicial					Durante BC			
			Dual	Primal	Hop-Prim	Var. Fix.	T(S)	Primal	Var. Fix.	T(S)	
TE20	3	449	446	457	496	453	0.01	449	87	0	
	4	385	384	385	385	924	0	-	0	0	
	5	366	361	366	366	1152	0	-	28	0	
TE30	3	597	594	609	609	898	0.02	597	53	0.01	
	4	521	515	522	541	1815	0	522	134	0	
	5	483	478	493	493	2191	0.02	483	84	0	
TE40	3	708	708	728	762	1401	0.02	-	2	0	
	4	627	623	630	676	2954	0.02	627	111	0	
	5	590	589	596	596	3935	0.03	590	223	0.01	
TE50	3	1366	1365	1375	1471	3223	0.03	1369	204	0.01	
	4	1212	1210	1259	1266	2724	0.11	1212	2036	0	
	5	1110	1103	1140	1149	4842	0.1	1110	1810	0.01	
TE60	3	1525	1521	1569	1714	2805	0.06	1525	1788	0.01	
	4	1336	1328	1373	1388	4207	0.15	1336	2795	0	
	5	1225	1225	1229	1279	9545	0.21	1225	36	0	
TE80	3	1806	1802	1840	1929	5970	0.13	1806	2387	0.02	
	4	1558	1549	1580	1601	9518	0.24	1558	3120	0.06	
	5	1442	1435	1477	1508	10912	0.39	1442	466	0.14	
TE100	3	2092	2082	2121	2329	9525	0.24	2092	4044	0.08	
	4	1788	1771	1888	2038	2000	1.02	1788	7769	0.12	
	5	1625	1625	1699	1760	9536	1.36	1625	15563	0.26	
TE120	3	1267	1258	1352	1424	308	0.8	1267	16912	0.16	
	4	1074	1071	1155	1197	728	1.48	1074	114	0.28	
	5	969	963	1020	1041	8449	2.02	969	1520	0.25	
TE160	3	1496	1488	1616	1761	4	5.49	1500	0	0.31	
	4	1229	1221	1286	1464	5704	10.4	1229	43756	0.88	
	5	1107	1098	1182	1259	1603	13.6	1107	925	1.71	

Tabela 2.8: Resultados das heurísticas do HMSTP para o grupo TE

apresentada em [17], para as instâncias usadas em [17]. Essas instâncias são esparsas e foram criadas a partir das instâncias de grafos completos, mencionadas no início desta seção. Foram usadas as instâncias de 40 e 80 vértices e foram escolhidas 400 e 800 arestas, respectivamente. A primeira coluna da tabela 2.9 indica o nome da instância, a segunda indica o número de arestas, a terceira o valor de H, e a quarta coluna indica o valor da solução ótima. Nas colunas seguintes, indicamos o valor da relaxação linear, a melhor solução encontrada e o tempo computacional para cada formulação.

Nome	E	H	Ótimo	Hopcut			HopMCF		
				LP	SOL	T(s)	LP	SOL	T(s)
TC40	400	3	566	566	566	0.06	566	566	1
		4	519	519	519	0.17	517.5	519	10
		5	496	496	496	0.46	494	496	177
TE40	400	3	710	710	710	0.39	701	710	1238
		4	625	625	625	1.36	619.5	625	28758
		5	581	581	581	0.41	578.5	581	33079
TR40	400	3	219	219	219	0.05	219	219	1
		4	176	176	176	0.07	176	176	4
		5	155	155	155	0.01	155	155	9
TC80	800	3	1054	1054	1054	3.78	1054	1054	28
		4	967	967	967	10.1	946.5	967	4634
		5	918	918	918	35.6	913.1	918	42003
TE80	800	3	1808	1808	1808	5.52	1794.1	1808	75765
		4	1568	1568	1568	60.4	1549.2	1568	> 2 semanas
		5	1442	1442	1442	304	1422.6	1751	-
TR80	800	3	208	208	208	0.1	208	208	4
		4	180	180	180	0.29	180	180	47
		5	164	164	164	0.84	164	164	115

Tabela 2.9: Comparando com a literatura do HMSTP

2.5 Comentários Finais

Apresentamos uma transformação do problema de árvore geradora mínima com restrição de nível (HMSTP) para o problema de Steiner em Grafos (PSG). Com essa transformação, mostramos que podemos usar modelos do problema de Steiner num grafo em nível para resolver o HMSTP. Apresentamos dois modelos para o PSG no grafo em nível; para o segundo modelo,

também apresentamos uma versão melhorada. Mostramos que esses modelos apresentam uma relaxação linear melhor do que as encontradas na literatura.

Para o primeiro modelo, foi proposto um esquema de *Branch-and-Cut* que é inicializado com uma heurística dual. Nesse algoritmo foram acrescentadas heurísticas primais e fixação de variáveis, onde uma das heurística foi proposta aqui. Os resultados obtidos com o algoritmo foram muito superiores aos encontrados na literatura. Resolvemos todas as instâncias da literatura em um tempo computacional significativamente inferior. As instâncias que estavam em aberto (não foram resolvidas de forma exata, só apresentaram limites para a instância) foram resolvidas da mesma forma que as instâncias em grafos completos até 160 vértices. Com isso, dobramos o tamanho da maior instância resolvida na literatura.

Capítulo 3

Árvore geradora mínima com restrição de diâmetro

3.1 Introdução

O problema de árvore geradora mínima com restrição de diâmetro, *Diameter constrained Minimum Spanning Tree Problem* (DMSTP), é definido como segue: dado um grafo $G = (V, E)$ com um conjunto de vértices $V = \{1, 2, \dots, n\}$, um conjunto de arestas $(i, j) \in E$ com custos associados c_{ij} e um número natural D , onde $2 \leq D \leq |V| - 1$, desejamos encontrar uma árvore geradora de custo mínimo, onde o caminho entre qualquer par de vértices tem no máximo D arestas.

Demonstrou-se que o DMSTP é NP-difícil para $D \geq 4$ em [30]. Algumas aplicações em redes de computadores, onde todos os nós da rede devem se comunicar com o menor custo possível e atingir um certo grau de qualidade, podem ser modeladas por esse problema (ver em [31]).

As formulações para o DMSTP na literatura usam implicitamente uma propriedade do problema que é a existência de um vértice central, quando D

é par, e uma aresta central, no caso de D ímpar. Essas formulações também usam uma estrutura de fluxo em rede para garantir a estrutura de árvore. Em [32, 33] foi proposta uma formulação de fluxo de um produto em rede que usa um vértice artificial para o caso par. Neste trabalho não foram feitos testes computacionais. GOUVEIA e MAGNANTI [34] propuseram uma formulação de fluxo de multiprodutos em rede, que apresenta um limite melhor que o apresentado em [32, 33], mas, devido ao grande número de variáveis, exige uso excessivo de memória. Para a formulação de Gouveia, foram feitos testes computacionais com grafos esparsos e obtidos resultados para instâncias de 60 vértices, para o caso de D par, e somente para problemas menores, no caso de D ímpar (40 vértices). Gouveia propôs outra formulação em [35] para o caso específico de D ímpar. Essa nova formulação apresentou uma relaxação linear um pouco inferior à apresentada em [34]. Entretanto, como possui um número menor de variáveis, foi possível resolver instâncias de até 60 vértices num grafo esparso. Em [36] foram propostas melhorias na formulação apresentada em [32, 33], que geraram bons resultados computacionais em grafos esparsos, de tamanho máximo de 60 vértices, para o caso de D ímpar, e de tamanho máximo de 40 vértices, para o caso de D par, e para grafos completos com no máximo 25 vértices. Uma formulação compacta 0 – 1 foi proposta em [37], usando a idéia de precedência de vértices. Os resultados computacionais obtidos em [37] foram comparados aos obtidos em [34, 36] e mostraram que nenhuma abordagem domina a outra. Uma nova abordagem usando *Constraint Programming* foi apresentada em [38] e o resultado foi superior, comparando com os resultados obtidos em [36, 37], em 25 instâncias, de um total de 29.

3.2 Formulação num grafo em níveis

Nesta seção, propomos uma adaptação da formulação para o HMSTP, apresentada na seção 2.2, para o DMSTP. Para isso, usaremos uma característica da solução do problema DMSTP: quando D for par, a árvore geradora mínima possui um vértice i central, que é ligado a duas subárvores de profundidade máxima $D/2$; quando D for ímpar, a árvore geradora mínima tem uma aresta (i, j) central, que é ligada a duas ou mais subárvores de profundidade máxima $(D - 1)/2$. Sendo assim, a grande diferença do problema DMSTP para o problema HMSTP é escolher a aresta central, para o caso ímpar, ou o vértice central, para o caso par. Para o caso par, é possível resolver o DMSTP, usando a solução para o HMSTP diretamente. Basta resolver $|V|$ problemas HMSTP, fixando um vértice raiz $i \in V$ em cada problema. Uma abordagem semelhante poderia ser usada em relação às arestas para o caso ímpar, mas não seria eficiente.

Assim como no HMSTP, construímos o grafo em nível. Para facilitar a notação, usaremos $\lfloor D/2 \rfloor = D/2$, quando D for par, e $\lfloor D/2 \rfloor = (D - 1)/2$, quando D for ímpar. Para o caso par, considere o grafo em níveis $G_P = (V_P, A_P)$, onde o conjunto de vértices V_P é definido como

$$V_P = \{0\} \cup \{(i, h) : 0 \leq h \leq \lfloor D/2 \rfloor, i \in V\}$$

e o conjunto de arcos A_P é definido como

$$\begin{aligned} A_P = & \{(0, (j, 0)) : j \in V\} \\ & \cup \{((i, h - 1), (j, h)) : (i, j) \in E, 1 \leq h \leq \lfloor D/2 \rfloor\} \\ & \cup \{((i, h), (i, \lfloor D/2 \rfloor)) : i \in V, 0 \leq h \leq \lfloor D/2 \rfloor - 1\}. \end{aligned}$$

O vértice 0 foi adicionado ao grafo para auxiliar na escolha do vértice central do problema original. O vértice (i, h) é associado ao vértice i no nível

h no grafo original, isto é, o caminho do vértice 0 até o vértice i contém $h + 1$ arestas. Note que o grafo G_P é construído em níveis com os vértices do grafo original copiados $\lfloor D/2 \rfloor$ vezes e o primeiro nível (zero) é ligado ao vértice raiz 0. Na figura 3.3 é apresentado um exemplo da transformação do grafo original (figura 3.1) para o grafo em níveis (figura 3.2), no caso de D par.

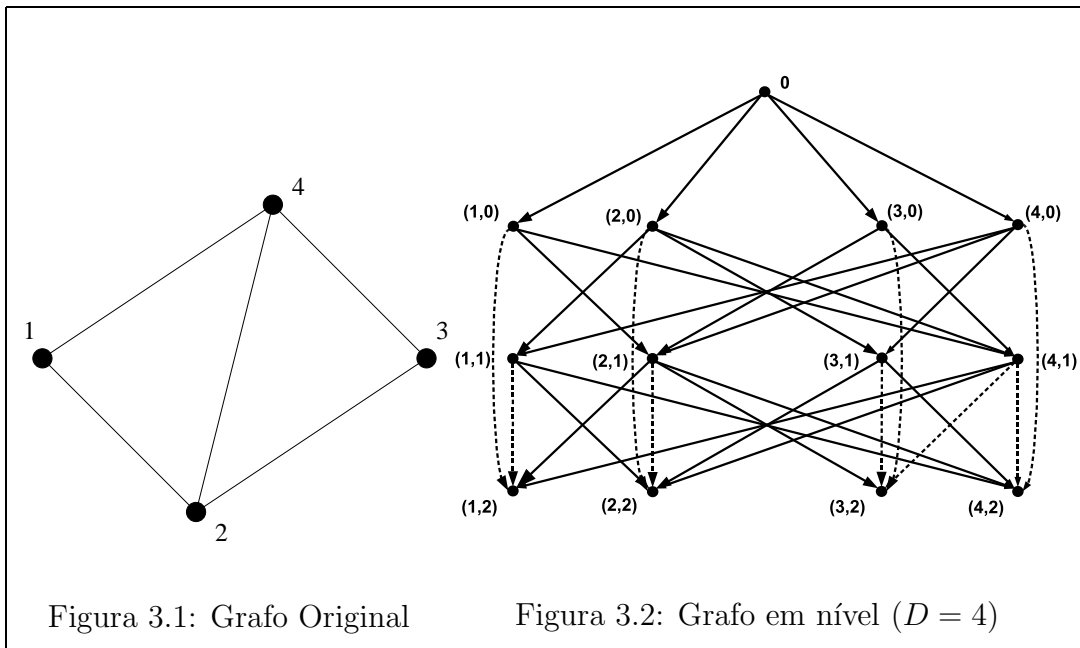


Figura 3.3: Exemplo da transformação do DMSTP para o PSG para D par

O grafo em níveis construído é o mesmo apresentado para o problema HMSTP. Sendo assim, podemos usar a mesma formulação de cortes orientados apresentada na seção 2.2.1. Para isso, adicionamos um custo alto \mathcal{C} aos arcos $(0, (j, 0))$, onde $j \in V$. Esse custo alto garante que somente um arco $(0, (j, 0))$ estará na solução, garantindo a escolha do vértice central (j).

Para o caso ímpar, considere o grafo em níveis $G_I = (V_I, A_I)$, onde o conjunto de vértices V_I é definido como

$$V_I = V_P \cup \{(i, -1) : i \in V\} \cup \{-1\}$$

e o conjunto de arcos A_I é definido como

$$\begin{aligned} A_I &= A_P \cup \{(i, 0), (j, -1) : (i, j) \in E\} \\ &\cup \{(i, -1), -1) : i \in V\} \\ &\cup \{(i, -1), (i, 0) : i \in V\}. \end{aligned}$$

Esse grafo é igual ao grafo G_P com a adição dos vértices -1 e $(i, -1)$, onde $i \in V$. Os vértices $(i, -1)$ foram adicionados para formar a aresta central do caso ímpar. Já o vértice -1 é adicionado para ser um vértice terminal e garantir que um vértice $(i, -1)$ estará na solução. Com isso, garantimos a construção do caminho do vértice raiz até o vértice terminal -1 , $\{(0, (j, 0)), ((j, 0), (i, -1)), ((i, -1), -1)\}$, e, implicitamente, estamos construindo a aresta central, porém, ainda temos que garantir que o vértice $(i, -1)$ volte a se conectar com o resto do grafo. Por isso, foram adicionados os arcos $((i, -1), (i, 0))$, onde $i \in V$. Para garantir que só um arco $((j, 0), (i, -1))$ estará na solução, utilizamos o mesmo artifício usado na escolha do arco $(0, (j, 0))$: adicionamos ao custo da aresta do grafo original c_{ji} um custo alto \mathcal{C} . Na figura 3.4 é apresentado um exemplo da transformação do grafo original (figura 3.1) para o grafo em níveis, no caso de D ímpar. Os arcos não mostrados na figura 3.4 estão em detalhe na figura 3.5.

Com esses grafos em níveis, podemos transformar esse problema em um problema de árvore de Steiner, como feito para o problema HMSTP. Para o caso de D par, os vértices terminais são $R_P = \{(i, \lfloor D/2 \rfloor) : i \in V\}$, e para o caso ímpar, os vértices terminais são $R_I = R_P \cup \{-1\}$. Não é difícil ver que a árvore geradora com diâmetro menor do que D no grafo original corresponde à árvore de Steiner em G_P , para o caso par, ou G_I , para o caso ímpar, com raiz no vértice 0 e contendo os vértices terminais R_P ou R_I , respectivamente.

O interessante dessa construção é que associando uma variável binária

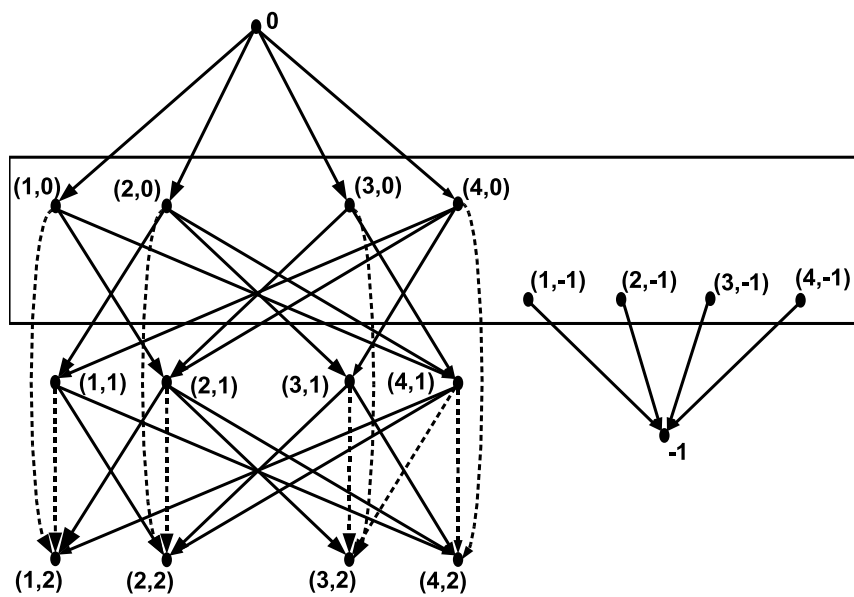


Figura 3.4: Exemplo da transformação do DMSTP para o PSG para D ímpar ($D = 5$)

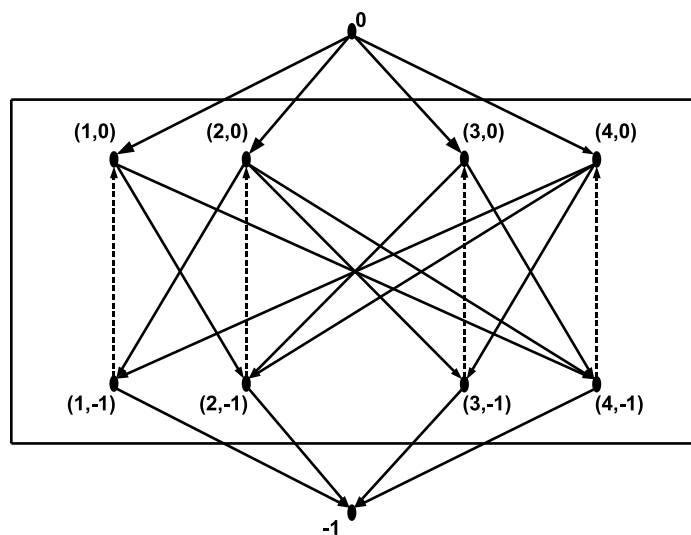


Figura 3.5: Detalhe das ligações entre os vértices no retângulo na figura 3.4

X_{ij}^h a cada arco em G_P , caso par, ou G_I , caso ímpar, podemos usar qualquer modelo para o problema de Steiner no grafo em níveis para fornecer um modelo válido para o DMSTP, como feito para o problema HMSTP na seção 2.2.

3.2.1 Formulação por cortes orientados

Para descrever os cortes orientados, usaremos a mesma notação apresentada na seção 2.2.1, alterando apenas o conjunto de vértices terminais que dependerá do valor de D , $R = R_P$, no caso par, e $R = R_I$, no caso ímpar.

Para o caso de D par, temos a seguinte formulação:

$$\min \mathcal{C} \sum_{j \in V} X_{0j}^{-1} + \sum_{(i,j) \in E} c_{ij} \sum_{h=1}^{\lfloor D/2 \rfloor} X_{ij}^h \quad (3.1)$$

$$\text{s.a.} \sum_{i \in V \setminus \{j\}} X_{ij}^{\lfloor D/2 \rfloor} + \sum_{h=1}^{\lfloor D/2 \rfloor} X_{jj}^h = 1 \quad j \in V \quad (3.2)$$

$$X^h(\delta(S)) \geq 1 \quad S \in \mathcal{S} \quad (3.3)$$

$$X_{0j}^{-1} \in \{0, 1\} \quad j \in V \quad (3.4)$$

$$X_{ij}^h \in \{0, 1\} \quad (i, j) \in E, h = 1, \dots, \lfloor D/2 \rfloor \quad (3.5)$$

$$X_{jj}^h \in \{0, 1\} \quad j \in V, h = 1, \dots, \lfloor D/2 \rfloor. \quad (3.6)$$

Para D ímpar, mantemos as restrições (3.2)-(3.6) e utilizamos a função objetivo a seguir:

$$\min \mathcal{C} \sum_{j \in V} X_{0j}^{-1} + \sum_{(i,j) \in E} c_{ij} \sum_{h=1}^{\lfloor D/2 \rfloor} X_{ij}^h + \sum_{(i,j) \in E} (c_{ij} + \mathcal{C}) X_{ij}^0$$

e adicionamos as seguintes restrições:

$$\sum_{i \in V} X_{i-1}^{-1} = 1 \quad (3.7)$$

$$X_{ij}^0 \in \{0, 1\} \quad (i, j) \in E \quad (3.8)$$

$$X_{i-1}^{-1} \in \{0, 1\} \quad i \in V \quad (3.9)$$

$$X_{ii}^0 \in \{0, 1\} \quad i \in V. \quad (3.10)$$

As restrições (3.2) (caso par), (3.2) e (3.7) (caso ímpar) garantem que cada vértice terminal seja visitado uma única vez. As restrições (3.3) são restrições de cortes orientados e garantem que a solução do problema possua pelo menos um arco em $\delta(S)$, o que torna a solução uma árvore de Steiner. A formulação descrita contém um número exponencial de restrições (3.3). O algoritmo de separação dessas restrições é o mesmo apresentado na seção 2.2.1.

3.3 Acelerando a resolução do modelo por cortes orientados

O procedimento de resolução é o mesmo apresentado na seção 2.3. Primeiro é aplicada a heurística dual *Dual Ascent*, em seguida, as heurísticas primais e, com posse de uma solução dual e primal, é possível tentar fixar variáveis pelo custo reduzido. Se não foi possível provar a otimalidade da melhor solução heurística encontrada, seguimos para a resolução do modelo exato com a adição dos cortes construídos na heurística dual.

Alguns ajustes foram necessários, o primeiro foi na escolha do componente-raiz da heurística *Dual Ascent*, no caso de D ímpar, e o segundo foi a substituição da heurística específica para o HMSTP, para uma adaptação dessa heurística para o DMSTP, detalhada na seção 3.3.1.

Para o caso de D ímpar, o vértice terminal -1 só é escolhido quando não existir nenhum outro componente-raiz, independentemente do método de escolha do componente-raiz (circular, MinArcos e MinArcos+MinSaturados).

Este procedimento vem de observações dos testes computacionais.

3.3.1 Heurística e Busca Local para o DMSTP

Como feito para o HMSTP, além das heurísticas para o PSG, também utilizamos uma heurística específica para o DMSTP. Essa heurística é, na verdade, uma adaptação da heurística Hop-Prim, detalhada na seção 2.3.4.

No caso de D par, só mudamos a primeira parte da heurística (construção de uma solução viável). A diferença é que escolhemos um subconjunto de vértices V_r , onde $V_r \subseteq V$, e executamos a heurística Hop-Prim $|V_r|$ vezes, cada uma com um vértice de V_r como raiz da árvore geradora mínima com restrição de nível com $H = \lfloor D/2 \rfloor$. A escolha do subconjunto V_r é feita de acordo com o grafo saturado G_π , construído pela heurística *Dual Ascent*. Para cada arco $(0, (j, 0))$ presente no grafo G_π , acrescentamos o vértice j no subconjunto V_r . Diferentemente do HMSTP, essa primeira parte da heurística modificada para o DMSTP não garante uma solução viável, pois não testamos todos os vértices como vértice central. A segunda parte da heurística Hop-Prim (busca local) só é executada para a melhor solução encontrada na primeira parte, se foi encontrada alguma.

No caso de D ímpar, a adaptação é muito parecida, com um procedimento a mais na primeira parte da heurística. Escolhemos, da mesma forma, um subconjunto de vértices v_r e executamos para cada vértice r de V_r a primeira parte da heurística Hop-Prim, construção de uma solução viável do problema de árvore geradora mínima com restrição de nível com $H = \lfloor D/2 \rfloor$ e raiz em r . Note que não levamos em consideração a aresta central, tratamos até o momento como no caso de D par. Se no final da primeira parte, para o vértice r , não foi possível construir uma árvore geradora, um novo procedimento é executado. Esse procedimento é uma maneira heurística de escolher uma

aresta central. Para cada aresta (r, j) presente na árvore gerada, verificamos se a adição de um nível para os filhos de j no nível $\lfloor D/2 \rfloor$ (vértices que têm $\lfloor D/2 \rfloor$ arestas no caminho único até o vértice r e esse caminho contém a aresta (r, j)) tornaria a árvore uma árvore geradora. Esse procedimento, na verdade, está escolhendo uma das arestas (r, j) como aresta central. Como no caso de D par, não garantimos uma solução viável. Se encontramos uma solução viável na primeira parte da heurística, executamos a segunda parte (busca local) para a melhor solução construída.

3.4 Resultados

Nesta seção, apresentamos nossos resultados computacionais e comparamos a formulação por cortes orientados com as melhores formulações e algoritmos da literatura para o DMSTP. As formulações escolhidas são apresentadas em [36, 34, 37, 38].

Para a comparação direta entre as formulações, usamos algumas das instâncias criadas em [36, 34]. Essas instâncias são as mais usadas para comparação. As instâncias criadas em [36] são de grafos completos de 15 a 25 vértices e de grafos esparsos de 20 a 40 vértices. Já as criadas em [34] são de grafos esparsos de 20 a 30 vértices e divididas em dois grupos (TR e TE). O grupo TR tem custo randômico e o grupo TE tem custo euclidiano. Para esses dois conjuntos de instâncias, resolvemos com os diâmetros de 4 a 8, como especificado nas referências. Para testar nossa formulação, usamos, também, instâncias do problema HMSTP. Essas instâncias são de grafos completos de 41 a 161 vértices. Podemos dividir essas instâncias em 3 grupos: dois grupos, TC e TE, com custo euclidiano, e o terceiro, TR, com custo randômico. Cada instância é resolvida com diâmetros de 4 a 12.

Os resultados computacionais foram obtidos em um PC Intel Core 2 Duo, 2.2 GHz com 2Gb de RAM, utilizando-se o programa XPRESS 2007A para resolver as relaxações lineares e a programação inteira, quando necessário. Comparamos nossos resultados com os resultados obtidos em [37, 38]. Os resultados computacionais apresentados em [37] foram obtidos em um PC Pentium IV 2.8 GHz com 2Gb de RAM, utilizando-se o programa CPLEX 8.1 para resolver as relaxações lineares e a programação inteira. Os resultados computacionais para os modelos propostos em [34, 36] também foram apresentados em [37]. Os resultados computacionais apresentados em [38] foram obtidos em um PC Pentium IV, 3 GHz com 1Gb de RAM, utilizando-se o programa OPL Studio 3.7.1 para resolver a *constraint programming*.

Os resultados obtidos com a formulação por cortes orientados para as instâncias do HMSTP são apresentados nas tabelas 3.1, 3.2 e 3.3. Cada tabela é referente aos resultados de um grupo de instâncias. A primeira coluna indica o valor de D . Nas colunas seguintes, informamos o nome da instância, o valor da solução ótima, o valor da relaxação linear, o número de nós necessários para encontrar a solução inteira ótima e o tempo computacional.

Como podemos notar nas tabelas 3.1, 3.2 e 3.3, a relaxação linear em quase todas as instâncias resolveu o problema. Quando foi necessário entrar no algoritmo *Branch-and-Cut*, o número de nós necessário para encontrar a solução ótima foi muito pequeno.

Os resultados das heurísticas estão nas tabelas 3.4, 3.5 e 3.6. A primeira coluna das tabelas indica o nome da instância, a segunda indica o valor de D e a terceira coluna indica o valor da solução ótima. Nas cinco colunas seguintes indicamos o melhor valor obtido com as heurísticas duais, a melhor solução encontrada por todas as heurísticas primais, a melhor solução encontrada pela heurística Hop-Prim adaptada, o número de variáveis fixadas pelo teste

D	Nome	Ótimo	LP	T(s)	Nós	Nome	Ótimo	LP	T(s)	Nós
4	TC40	747	747	0.27	1	TC60	1083	1081.5	1.32	5
5		673	673	0.38	1		977	977	3.98	1
6		606	606	4.49	1		856	856	6.86	1
7		575	575	2.14	1		821	821	5.96	1
8		544	544	6.05	1		781	781	386	1
9		532	532	2.22	3		760	760	9.87	1
10		516	516	9.73	1		734	734	91.8	1
11		508	508	0.72	1		722	722	5.91	1
12		498	498	6.55	1		712	712	7.38	1
4	TC80	1303	1303	1.39	1	TC100	1549	1549	9.58	1
5		1203	1203	8.36	1		1438	1438	362	1
6		1064	1064	226	1		1259	1259	1858	1
7		1024	1024	169	1		1220	1220	1333	1
8		976	976	2933	1		1158	1158	4574	1
9		952	952	1869	1		1136	1136	522	1
10		920	920	2341	1		1104	1104	31718	1
11		906	906	2510	1		1088	1088	42182	1
12		884	884	776	1		1060	1060	7416	1
4	TC120	1431	1431	6.82	1	TC160	1731	1731	19.7	1
5		1281	1281	42.7	1		1572	1572	8377	1
6		1059	1059	1183	1		1247	1245.83	13611	11
7		1005	1005	322	1		1177	1177	15560	1
8		925	925	4023	1		-	-	-	-
9		894	894	5636	1		-	-	-	-
10		851	851	13901	1		-	-	-	-
11		836	836	36925	1		-	-	-	-
12		812	812	29596	1		-	-	-	-

Tabela 3.1: Resultados do modelo para o DMSTP para o grupo TC

D	Nome	Ótimo	LP	T(s)	Nós	Nome	Ótimo	LP	T(s)	Nós
4	TE40	742	742	0.31	1	TE60	1657	1657	0.68	1
5		678	678	0.76	1		1528	1528	2.72	1
6		606	606	0.43	1		1317	1317	25.1	2
7		585	585	1.19	1		1255	1255	15.9	1
8		562	562	2.54	1		1174	1174	6.16	1
9		553	552.5	53.1	1		1139	1139	44.1	1
10		537	537	0.15	1		1083	1083	1.04	1
11		529	529	1.52	1		1063	1063	2.89	1
12		525	525	30.7	1		1044	1044	21.1	1
4	TE80	2045	2045	2.11	1	TE100	2377	2377	4.07	1
5		1828	1828	6.39	1		2166	2166	217	1
6		1564	1564	12.8	1		1802	1802	257	1
7		1479	1479	39.8	1		1708	1708	352	1
8		1399	1399	963	1		1585	1585	1037	2
9		1355	1355	3786	1		1545	1545	428	1
10		1293	1293	20.4	1		1486	1486	879	1
11		1267	1267	30.1	1		1453	1453	1302	1
12		1232	1232	309	1		1415	1415	5540	1
4	TE120	1448	1448	6.72	1	TE160	1741	1741	23.8	1
5		1297	1297	4.31	1		1583	1583	6918	1
6		1077	1077	2120	1		1253	1251.83	14422	11
7		1019	1019	333	1		1182	1182	20812	1
8		938	938	4213	1		1081	1081	158572	1
9		907	907	415	1		-	-	-	-
10		866	866	9565	1		-	-	-	-
11		848	848	4859	1		-	-	-	-
12		826	826	74178	2		-	-	-	-

Tabela 3.2: Resultados do modelo para o DMSTP para o grupo TE

D	Nome	Ótimo	LP	T(s)	Nós	Nome	Ótimo	LP	T(s)	Nós		
4	TR40	249	249	0.05	1	TR60	264	264	0.15	1		
5		198	198	0.43	1		213	211.25	3.34	5		
6		155	155	0.24	1		155	155	7.47	1		
7		144	144	0.75	1		140	140	25.1	1		
8		135	135	0.9	1		124	124	7.12	1		
9		131	131	0.48	1		117	117	8.75	1		
10		129	129	1.04	1		114	114	8.32	1		
11		128	128	2.08	1		112	111.5	13.7	1		
12		127	127	0.22	1		108	108	0.57	1		
4		TR80	416	416	0.93		1					
5			326	324	884		15					
6			208	208	85.8		1					
7	187		187	91.3	1							
8	168		167.25	8.41	1							
9	160		160	14.7	1							
10	153		153	2.39	1							
11	152		152	2.41	1							
12	151		151	1.65	1							

Tabela 3.3: Resultados do modelo para o DMSTP para o grupo TR

de custo reduzido e o tempo computacional total do procedimento inicial (heurísticas duais e primais, e fixação de variáveis). As últimas três colunas são referente à heurística primal, ao número de variáveis fixadas e ao tempo computacional total das heurísticas e fixações de variáveis durante o BC.

Na tabela 3.7, o desempenho da formulação por cortes orientados (Hopcut) é comparado com as formulações apresentadas em [34, 37], usando as instâncias criadas em [34]. As formulações apresentadas em [34, 37] serão denominadas aqui por G&M e ILP, respectivamente. A primeira coluna da tabela 3.7 indica o número de vértices da instância, e a segunda, o número de arestas. A terceira coluna indica o valor de D . Nas colunas seguintes indicamos o tempo computacional para cada formulação.

Na tabela 3.8, o desempenho da formulação por cortes orientados (Hopcut) é comparado com as formulações apresentadas em [36, 37, 38], usando as instâncias criadas em [36]. As formulações apresentadas em [36, 38] serão denominadas aqui por LiftDMST e CP, respectivamente. A primeira coluna da tabela 3.8 indica o número de vértices da instância, e a segunda, o número de arestas. A terceira coluna indica o valor de D . Nas colunas seguintes

Nome	D	Ótimo	Procedimento Inicial				Durante BC			
			Dual	Primal	Hop-Prim	Var. Fix.	T(S)	Primal	Var. Fix.	T(S)
TC40	4	747	735	753	802	2375	0.02	749	504	0
	5	673	673	740	759	907	0.1	673	2740	0
	6	606	593	640	640	1961	0.07	606	68	0.02
	7	575	568	597	597	3208	0.09	575	2194	0.02
	8	544	536	587	587	2261	0.14	548	3344	0.06
	9	532	530	536	540	6946	0.13	532	301	0.01
	10	516	506	526	538	6187	0.13	516	1639	0
	11	508	508	513	513	8444	0.15	508	422	0.02
	12	498	494	500	502	8626	0.22	498	870	0
TC60	4	1083	1065	1084	1200	5794	0.03	1083	953	0.01
	5	977	968	1005	1082	6199	0.15	977	2392	0.05
	6	856	849	919	933	2498	0.26	856	7603	0.03
	7	821	819	859	859	6538	0.35	821	5822	0.02
	8	781	769	828	828	4273	0.47	781	766	0.18
	9	760	757	801	801	7349	0.66	760	8790	0.12
	10	734	729	775	783	8092	0.61	734	7674	0.09
	11	722	722	732	732	18727	0.58	722	139	0.02
	12	712	712	720	728	20239	0.39	712	1301	0.04
TC80	4	1303	1292	1316	1426	10208	0.05	1303	2088	0.01
	5	1203	1199	1302	1302	2549	0.89	1203	12053	0.16
	6	1064	1056	1124	1124	5128	0.63	1064	699	0.09
	7	1024	1016	1084	1084	4920	1.27	1024	15669	0.44
	8	976	965	1032	1032	4990	1.14	976	6117	0.32
	9	952	947	980	980	16820	1.42	952	1079	0.14
	10	920	916	956	956	17234	1.83	920	8565	0.51
	11	906	902	934	937	21782	1.68	906	3612	0.53
	12	884	881	928	928	17675	2.26	884	17777	1.35
TC100	4	1549	1529	1600	1799	9920	0.18	1552	8686	0.06
	5	1438	1430	1569	1694	2266	2.35	1438	12925	0.37
	6	1259	1245	1337	1337	2925	1.52	1259	1186	0.32
	7	1220	1216	1291	1291	4186	2.72	1220	28888	0.6
	8	1158	1155	1223	1223	5684	2.58	1158	1020	0.23
	9	1136	1136	1193	1193	10597	3.59	1136	32595	0.69
	10	1104	1098	1124	1124	34190	2.28	1104	6419	1.58
	11	1088	1084	1126	1133	25570	4.06	1088	15480	1.15
	12	1060	1060	1104	1104	30242	3.68	1060	25745	1
TC120	4	1431	1417	1456	1656	20301	0.26	1431	2126	0.04
	5	1281	1281	1559	1559	680	6.58	1281	34811	0.27
	6	1059	1043	1139	1139	6776	2.73	1059	431	0.13
	7	1005	1003	1152	1152	42	6.35	1005	48664	0.78
	8	925	916	1026	1026	1065	4.95	925	28	0.33
	9	894	892	1004	1004	159	6.87	894	6	0.53
	10	851	845	943	943	1079	6.99	851	246	0.61
	11	836	830	906	906	8578	9.83	836	1245	1.08
	12	812	807	862	862	24605	8.43	812	2239	1.54
TC160	4	1731	1718	1753	2066	37909	0.5	1731	8503	0.06
	5	1572	1555	1661	1989	12586	8.2	1572	13317	0.17
	6	1247	1227	1394	1448	2654	7.65	1247	68794	0.65
	7	1177	1173	1358	1388	159	16.1	1177	0	0.62
	8	-	1062	1179	1240	504	12.2	-	-	-
	9	-	1037	1110	1135	13016	18.1	-	-	-
	10	-	982	1055	1065	15391	18.2	-	-	-
	11	-	963	1010	1078	51812	19.4	-	-	-
	12	-	934	977	1041	66593	18.6	-	-	-

Tabela 3.4: Resultados das heurísticas do DMSTP para o grupo TC

Nome	D	Ótimo	Procedimento Inicial				Durante BC			
			Dual	Primal	Hop-Prim	Var. Fix.	T(S)	Primal	Var. Fix.	T(S)
TE40	4	742	730	763	787	1913	0.02	742	1050	0.01
	5	678	670	715	715	1831	0.06	678	1930	0.02
	6	606	601	620	637	3660	0.04	609	789	0
	7	585	579	600	625	4203	0.08	585	1315	0.01
	8	562	555	574	589	4784	0.09	562	1194	0.03
	9	553	544	566	582	5142	0.12	553	564	0.08
	10	537	537	537	551	-	0.15	-	-	-
	11	529	527	537	557	7920	0.17	529	922	0.01
	12	525	522	533	533	8307	0.21	-	440	0.01
TE60	4	1657	1649	1682	1894	5779	0.07	1657	1283	0
	5	1528	1521	1592	1754	4888	0.26	1528	2218	0.02
	6	1317	1295	1398	1398	3160	0.31	1317	484	0
	7	1255	1243	1372	1372	1352	0.76	1255	10788	0.12
	8	1174	1167	1220	1247	9118	0.63	1175	5247	0.04
	9	1139	1131	1237	1237	3610	1.06	1139	11953	0.16
	10	1083	1083	1083	1178	-	1.04	-	-	-
	11	1063	1063	1109	1168	14368	0.96	1063	5078	0.05
	12	1044	1038	1066	1127	17761	0.45	1055	2169	0.06
TE80	4	2045	2024	2068	2127	10356	0.08	2045	2180	0.02
	5	1828	1811	1843	2090	13408	0.22	1828	2350	0.02
	6	1564	1554	1718	1718	1658	0.79	1564	16965	0.07
	7	1479	1468	1644	1644	652	1.56	1479	21225	0.12
	8	1399	1385	1544	1544	1041	2.25	1399	12308	0.22
	9	1355	1334	1502	1502	511	2.33	1355	272	0.64
	10	1293	1293	1340	1397	22333	1.9	1293	8872	0.15
	11	1267	1265	1322	1343	22869	2.53	1267	10442	0.4
	12	1232	1219	1246	1281	33505	1.51	1232	4210	0.32
TE100	4	2377	2351	2433	2671	12838	0.19	2377	6729	0.06
	5	2166	2142	2231	2542	12498	0.9	2166	10264	0.05
	6	1802	1782	1934	2024	4872	1.29	1802	22806	0.1
	7	1708	1698	1916	1916	337	4.09	1708	1	0.18
	8	1585	1565	1622	1717	24930	1.55	1585	2304	0.24
	9	1545	1541	1650	1650	9126	5.27	1545	35015	0.47
	10	1486	1484	1545	1599	26042	5.28	1486	2425	0.23
	11	1453	1450	1539	1539	18031	6.24	1453	31778	0.26
	12	1415	1411	1464	1518	37427	4.95	1415	1949	0.2
TE120	4	1448	1436	1468	1632	22127	0.25	1448	6002	0.01
	5	1297	1297	1298	1527	36488	2.39	1297	132	0
	6	1077	1060	1197	1203	1967	2.91	1077	0	0.19
	7	1019	1019	1098	1098	5525	6.3	1019	41928	0.31
	8	938	929	1021	1021	3086	5.34	938	225	0.33
	9	907	907	999	1008	1891	7.76	907	29	0.29
	10	866	857	961	961	1095	7.03	866	31	0.56
	11	848	844	905	905	16818	9.82	848	56333	0.43
	12	826	817	853	904	46490	7.88	826	16929	0.83
TE160	4	1741	1729	1751	2100	43400	0.38	1741	6830	0.03
	5	1583	1563	1726	1924	6334	11.3	1583	0	0.16
	6	1253	1231	1449	1449	434	7.98	1253	72801	0.64
	7	1182	1174	1417	1417	6	15	1182	0	2.03
	8	1081	1064	1226	1226	3	12.7	1081	0	1.55
	9	-	1035	1271	1271	0	17.8	-	-	-
	10	-	986	1095	1095	665	19.6	-	-	-
	11	-	961	1002	1089	59736	29.3	-	-	-
	12	-	934	958	1029	107686	11.96	-	-	-

Tabela 3.5: Resultados das heurísticas do DMSTP para o grupo TE

Nome	D	Ótimo	Procedimento Inicial				Durante BC			
			Dual	Primal	Hop-Prim	Var. Fix.	T(S)	Primal	Var. Fix.	T(S)
TR40	4	249	244	249	249	3318	0.01	-	0	0
	5	198	185	204	282	3409	0.04	198	493	0
	6	155	151	155	168	4723	0.04	-	160	0.01
	7	144	140	145	166	5197	0.05	144	472	0
	8	135	134	139	139	6061	0.06	135	313	0
	9	131	131	134	134	7034	0.09	131	88	0
	10	129	128	132	133	7668	0.08	129	388	0.01
	11	128	126	128	128	8422	0.13	-	449	0.01
12	127	127	130	130	9281	0.1	127	35	0	
TR60	4	264	261	264	266	7144	0.03	-	14	0
	5	213	200	217	274	7529	0.07	213	1049	0.02
	6	155	148	165	202	8670	0.1	155	1345	0.04
	7	140	134	166	166	8530	0.21	140	3393	0.12
	8	124	122	130	138	12932	0.16	124	296	0.04
	9	117	116	125	131	14304	0.22	117	1422	0.01
	10	114	113	116	118	16850	0.22	114	494	0.04
	11	112	110	113	113	18762	0.29	112	498	0.03
12	108	107	108	109	21444	1	-	33	0	
TR80	4	416	402	418	423	12002	0.05	416	406	0.02
	5	326	298	331	395	12092	0.22	326	1959	0.07
	6	208	196	228	242	14747	0.28	208	1686	0.02
	7	187	182	198	234	19417	0.35	187	649	0.07
	8	168	166	171	176	25016	0.36	168	666	0.06
	9	160	159	170	175	26456	0.43	160	2380	0.06
	10	153	153	157	163	31525	0.49	153	788	0
	11	152	152	155	155	34776	0.55	152	270	0.01
12	151	151	153	154	37979	0.59	151	168	0	

Tabela 3.6: Resultados das heurísticas do DMSTP para o grupo TR

Nome	$ E $	D	Hopcut T(s)	G&M T(s)	ILP T(s)
TR20	100	4	0	0.5	0.9
		5	0	6.3	2.4
		6	0.05	5.8	2.9
		7	0.08	94	2.6
		8	0.01	1.3	3.4
TR30	200	4	0	0.8	3.5
		5	0.01	58.6	283
		6	0.09	2.9	5.7
		7	0.15	529	53.9
		8	0.07	2.3	3.67
TE20	100	4	0.05	0.1	1.1
		5	0	5.3	1.7
		6	0	3.1	7.3
		7	0.03	49.5	10
		8	0	1.1	10.7
TE30	200	4	0.07	130	59.5
		5	0.1	25.1	36.1
		6	0.08	1381	348
		7	0.4	6912	1014
		8	0.21	1111	2430

Tabela 3.7: Resultados dos modelos para o DMSTP para as instâncias criadas por Gouveia

indicamos o tempo computacional para cada formulação.

$ V $	$ E $	D	Hopcut T(s)	LiftDMST T(s)	ILP T(s)	CP T(s)
15	105	4	0.05	4.7	0.7	0.08
		5	0.06	22.8	3	0.22
		6	0.06	18.6	8.1	0.28
		7	0.07	26.9	20	0.38
		9	0.07	6.2	10.7	0.47
		10	0.08	1	4.3	0.41
20	190	4	0.07	562	2.5	0.2
		5	0.08	436	8.1	1.06
		6	0.06	455	95	2.03
		7	0.01	5.1	4.5	0.97
		9	0.01	73.4	66.7	5.01
		10	0.01	29.7	101	6.08
25	300	4	0	15203	12	1.48
		5	0.07	>20000	64.3	2.83
		6	0.11	826	26.4	39.1
		7	0.13	11521	770	56
		9	0.11	246	295	114
		10	0.11	254	404	55.4
20	50	4	0.03	1	0.2	0.05
		5	0	4.6	1	0.17
		6	0	0.8	5.1	0.13
		7	0.06	0.8	1.2	0.14
		9	0.01	0.7	2.8	0.45
		10	0	0.2	1.3	0.64
40	100	4	0.07	43.7	1.9	5.44
		5	0.08	291	6.4	7.31
		6	0	50.9	13.2	4.72
		7	0.09	459	212	34.3
		9	0.13	1565	979	40.1

Tabela 3.8: Resultados dos modelos para o DMSTP para as instâncias criadas por Santos

Como podemos ver nas tabelas 3.7 e 3.8, os resultados obtidos com a nova formulação Hopcut são superiores aos anteriores. Em todas as instâncias, o tempo necessário para resolver foi inferior a um segundo. Também podemos notar que quanto maior for o tamanho das instâncias, maior será a diferença de desempenho. Todas as instâncias foram resolvidas só com a relaxação linear.

3.5 Comentários Finais

Como feito para o HMSTP, apresentamos uma transformação do problema de árvore geradora mínima com restrição de diâmetro (DMSTP) para o problema de Steiner em Grafos (PSG). Com essa transformação, mostramos que podemos usar modelos do problema de Steiner num grafo em nível para resolver o DMSTP. Apresentamos um modelo para o PSG no grafo em nível.

Os resultados obtidos com o algoritmo proposto aqui foram bem superiores aos encontrados na literatura. Resolvemos as instâncias da literatura em um tempo computacional significativamente inferior e, pela primeira vez, resolvemos de forma exata instâncias de grafos completos, superiores a 25 vértices, chegando até 160 vértices. Como para o HMSTP, a diferença entre o valor da relaxação linear e a solução ótima foi muito pequena. Em quase todas as instâncias foi possível resolver o problema só com a relaxação linear. Nas instâncias em que foi necessário entrar no método *Branch-and-Cut*, encontramos a solução ótima inteira resolvendo poucos nós (no máximo 15).

Capítulo 4

Árvore geradora com número máximo de folhas

4.1 Introdução

O problema de árvore geradora com número máximo de folhas, *Maximum Leaf Spanning Tree Problem* (MLSTP), é definido como segue: seja $G = (V, E)$ um grafo conexo não-direcionado, formado por um conjunto V de vértices e um conjunto E de arestas. Denomina-se uma árvore de G a qualquer um dos subgrafos conexos acíclicos daquele grafo. Árvores de G , por sua vez, são chamadas de geradoras quando contêm todos os vértices em V . Seja $\delta(i) \subseteq E$ o conjunto das arestas com uma extremidade em $i \in V$. O número $|\delta(i)|$ dessas arestas é denominado o grau de i . Em particular, em uma árvore de G , vértices com grau 1 são denominados folhas, e o problema da árvore geradora com número máximo de folhas consiste em encontrar uma árvore geradora de G com o maior número possível de folhas.

Esse problema modela, dentre outras aplicações práticas, projetos de redes de telecomunicações e o desenho de layouts de circuitos eletrônicos

[39, 40]. Por exemplo, considere o caso de redes de telecomunicação onde os vértices correspondem a terminais e o objetivo é projetar uma rede com arquitetura de uma árvore. “Terminais folhas” envolvem então uma menor carga de trabalho que os “terminais intermediários”, com grau pelo menos dois, que podem eventualmente servir como ponto intermediário no roteamento de mensagens. Nessas aplicações (vide [41] para maiores informações), maximizar o número de folhas se mostra, em diversas situações práticas, um objetivo bastante “atraente”.

Embora seja de solução trivial, se G é um grafo completo, o MLSTP é, no caso geral, quando G é esparso, um problema NP-difícil [30]. Um algoritmo de solução, com um fator de aproximação 3, foi proposto para o problema em [42, 43]. Posteriormente, outro algoritmo, com um fator de aproximação 2, foi sugerido em [44]. A seguir, foi demonstrado em [45] que MLSTP é MAX-SNP-difícil, o que implica em dizer que existe um $\epsilon > 0$ tal que é NP-difícil encontrar uma solução com fator de aproximação $(1 + \epsilon)$, para o problema.

Duas formulações e um algoritmo de solução exata foram propostos para o MLSTP em [41], sendo lá aplicados a uma dada variante do problema. Essa variante considera custos $\{c_e : e \in E\}$ para as arestas de G e impõe um número fixo pré-especificado de folhas a aparecer na árvore geradora de custo mínimo que se deseja encontrar. O algoritmo proposto em [41], do tipo *Branch-and-Bound*, utiliza Relaxação Lagrangeana para gerar limites duais, obtidos através da resolução de árvores geradoras de custo mínimo. Instâncias aleatórias do problema, com até 40 vértices, foram resolvidas em [41], com garantia de otimalidade. Duas outras formulações e um estudo poliedral a elas associado, foram introduzidos em [46]. Um algoritmo *Branch-and-Bound*, baseado em uma dessas duas formulações, foi implementado em [47]. Esse algoritmo, que não utiliza nenhuma das desigualdades válidas for-

tes, caracterizadas em [46], gera limites superiores para o MLSTP, através da resolução de um problema da árvore geradora de custo mínimo, convenientemente definido sobre G . O algoritmo exato em [47] parece ser o único, na literatura, especificamente desenvolvido para o MLSTP. Esse algoritmo foi capaz de resolver, com garantia de otimalidade, instâncias aleatórias do MLSTP com até 100 vértices.

4.2 Uma formulação da literatura

Nesta seção, apresentamos uma das duas formulações propostas em [46] e, em seguida, o algoritmo do tipo *Branch-and-Bound* implementado em [47].

Associe variáveis $x = \{x_e \in \mathbb{R}_+ : e \in E\}$ e $z = \{z_i \in \{0, 1\} : i \in V\}$, respectivamente, às arestas e aos vértices de G . Variáveis no primeiro conjunto definem a árvore que se quer construir, enquanto aquelas, no segundo, explicitam os vértices que induzem folhas daquela árvore. Uma formulação para o MLSTP [46] é então dada por

$$\max \sum_{i \in V} z_i \quad (4.1)$$

$$\text{s.a. } \sum_{e \in E} x_e = |V| - 1 \quad (4.2)$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \quad S \subset V, |S| \geq 2 \quad (4.3)$$

$$\sum_{e \in \delta(i)} x_e + (|\delta(i)| - 1)z_i \leq |\delta(i)| \quad i \in V \quad (4.4)$$

$$0 \leq x_e \leq 1 \quad e \in E \quad (4.5)$$

$$z_i \in \{0, 1\} \quad i \in V. \quad (4.6)$$

Na desigualdade (4.3), $E(S)$ são todas as arestas $(i, j) \in E$ onde $i \in S$ e $j \in S$. Nessa formulação, as desigualdades (4.2), (4.3) e (4.5) definem o politopo da árvore geradora, como caracterizado por EDMONDS [48]. Pelas

desigualdades (4.4), quando um vértice $i \in V$ induz uma folha da árvore de solução, $z_i = 1$ explicita esse fato. Nesse caso, em consequência, apenas uma das arestas em $\delta(i) \subseteq E$ poderá também aparecer na árvore. Caso contrário, se i não induz uma folha da árvore de solução, $z_i = 0$ se impõe, e todas as arestas em $\delta(i)$ estarão eventualmente livres para aparecer na mesma. Vale ressaltar que são precisamente as restrições (4.4) que fazem com que essa formulação seja uma versão fortalecida de uma formulação para o MLSTP, introduzida em [41].

Como observado em [46], para que o vetor z represente exatamente o conjunto de folhas da árvore geradora que se deseja obter, as desigualdades

$$\sum_{e \in \delta(i)} x_e + z_i \geq 2 \quad i \in V \quad (4.7)$$

devem ser adicionadas à formulação. Note, entretanto, que, devido à função objetivo (4.1), onde todos os coeficientes são positivos, as desigualdades acima são naturalmente satisfeitas pela solução inteira a ser obtida. De qualquer forma, essas desigualdades são válidas para aquela formulação e podem ser utilizadas para reforçar a relaxação linear da mesma.

Algumas desigualdades adicionais que, sob condições não muito fortes, definem facetas do politopo associado a (4.2)-(4.6), foram também caracterizadas em [46]. Em particular, a família de desigualdades

$$\sum_{e \in F} x_e + (|F| - 1)z_i \leq |F| \quad i \in V, F \subseteq \delta(i), |F| \geq 2 \quad (4.8)$$

pode ser utilizada para fortalecer a relaxação linear daquela formulação.

4.2.1 Um algoritmo exato

A formulação (4.1)-(4.6) pode ser reescrita como

$$\max \sum_{i \in V} z_i \quad (4.9)$$

$$\text{s.a. } \sum_{e \in \delta(i)} x_e + (|\delta(i)| - 1)z_i \leq |\delta(i)| \quad i \in V \quad (4.10)$$

$$x \in ST_G \quad (4.11)$$

$$z_i \in \{0, 1\} \quad i \in V, \quad (4.12)$$

onde ST_G é a região poliedral definida por (4.2), (4.3) e (4.5), ou seja, uma árvore geradora de G .

Como (4.10) e (4.11) garantem que $z_i \leq 1$ para $i \in V$, podemos relaxar linearmente a restrição (4.12) como uma restrição de não-negatividade. Então, numa solução ótima do problema relaxado, a restrição (4.10) deve ser satisfeita como igualdade. Com isso, o problema relaxado é equivalente ao seguinte problema:

$$\max \quad \sum_{i \in V} \frac{|\delta(i)| - \sum_{e \in \delta(i)} x_e}{|\delta(i)| - 1} \quad (4.13)$$

$$= \sum_{i \in V} \frac{|\delta(i)|}{|\delta(i)| - 1} - \sum_{e=(i,j) \in E} \left(\frac{1}{|\delta(i)| - 1} + \frac{1}{|\delta(j)| - 1} \right) x_e$$

$$\text{s.a. } x \in ST_G. \quad (4.14)$$

Podemos resolver esse problema eficientemente, usando um algoritmo para árvore geradora mínima ([1]), com os custos das arestas $e = (i, j) \in E$ igual a $1/(|\delta(i)| - 1) + 1/(|\delta(j)| - 1)$.

O algoritmo *Branch-and-Bound* descrito em [47] é baseado na formulação (4.9)-(4.12) e utiliza o problema relaxado (4.13)-(4.14) para gerar limites superiores válidos, em cada nó da árvore de procura. Denotaremos por $SP(S_1, S_0, F)$ o subproblema genérico em qualquer um desses nós, onde (S_1, S_0, F) é uma partição de V , onde qualquer vértice em S_1 tem que ser folha, nenhum vértice de S_0 pode ser folha, e $F = V \setminus \{(S_1 \cup S_0)\}$ é o conjunto de vértices livres. No nó raiz, o subproblema corresponde ao problema original, que é expresso por $SP(\emptyset, \emptyset, V)$.

A relaxação linear associada ao subproblema $SP(S_1, S_0, F)$ é dada por

$$\max \sum_{i \in F} \frac{|\delta(i)|}{|\delta(i)| - 1} - \sum_{e=(i,j) \in E} d_e x_e + |S_1| \quad (4.15)$$

$$\text{s.a.} \quad \sum_{e \in \delta(i)} x_e \leq 1 \quad i \in S_1 \quad (4.16)$$

$$x \in ST_G, \quad (4.17)$$

onde

$$d_e = \begin{cases} \frac{1}{|\delta(i)|-1} + \frac{1}{|\delta(j)|-1} & i, j \in F, \\ \frac{1}{|\delta(i)|-1} & i \in F, j \notin F, \\ 0 & i, j \notin F, \end{cases}$$

e pode também ser resolvido por um algoritmo que encontre uma árvore geradora mínima para $G \setminus S_1$, sob custos d_e , conectando-se a seguir os vértices de S_1 àquela árvore.

O algoritmo Branch-and-Bound proposto em [47], é esquematicamente descrito em 4.1, a seguir.

4.2.2 Detalhes da implementação

Para efetuar comparações com os métodos aqui propostos, implementamos o algoritmo 4.1. Após a realização de alguns testes, identificamos algumas possibilidades de melhora naquele algoritmo.

A primeira modificação é feita no primeiro passo do algoritmo (Iniciação). Verificamos que o algoritmo aproximativo Solis-Obi [44] não apresentou bons resultados e o eliminamos da nossa implementação (comportamento similar foi também observado em [47]).

No segundo passo do algoritmo (Seleção do Subproblema), criamos uma nova opção. Além de escolher um subproblema através da estratégia *depth-first* (o nó mais profundo da árvore de busca é o escolhido), implementamos

Algoritmo 4.1 Branch-and-Bound

1 - Iniciação: Executar as seguintes heurísticas:

BFS: Gerar árvores geradoras com raiz em $i \in V$ (usou um algoritmo de busca *breadth first* [49]).

Lu-Ravi: Executar algoritmo 3-aproximado [43].

Solis-Obi: Executar algoritmo 2-aproximado [44].

Seja LB a melhor solução encontrada (número de folhas).

Se $LB = |V| - 1$ então FIM se não $L := \{SP(\emptyset, \emptyset, V)\}$

2 - Seleção do Subproblema:

Se $L = \{\emptyset\}$ então FIM

Escolha $SP(S_1, S_0, F) \in L$ de maneira *depth-first*.

Se $|S_1| + |F| < LB$ então Volte para 2.

3 - Checando viabilidade:

Se $SP(S_1, S_0, F)$ não é viável então Volte para 2.

4 - Atualizando o limite inferior:

Se $|S_1| > LB$ então

Execute *BFS* com raiz em $v \in S_0 \cup F$ e fazendo os vértices de S_1 folhas.

Seja LB a nova melhor solução.

fim Se

5 - Limite superior:

Resolva o problema (4.15)-(4.17) e compute UB

Se $\lfloor UB \rfloor \leq LB$ então Volte para 2.

6 - Criação dos Subproblemas:

$L := L \cup \{SP(S_1, \bar{S}_0, \bar{F}), SP(\bar{S}_1, S_0, \bar{F})\}$, onde $v = \operatorname{argmax}\{|\delta_G(u)| : u \in F\}$, $\bar{S}_1 = S_1 \cup \{v\}$, $\bar{S}_0 = S_0 \cup \{v\}$ e $\bar{F} = F \setminus \{v\}$.

Volte para 2.

também uma opção onde o escolhido é o subproblema de maior limite superior (*best bound*). Um benefício direto dessa modificação foi o fortalecimento do teste de poda $|S_1| + |F| < \text{LB}$ para $|S_1| + |F| \leq \text{LB}$.

O quarto passo (atualização do limite inferior) do algoritmo 4.1 só ocorre se $|S_1| > \text{LB}$. Na nossa implementação, além de atualizar o limite inferior nesse quarto passo, também atualizamos o limite inferior em todas as iterações onde um limite superior é obtido. Note que isso é possível pois no cálculo do limite superior construímos uma árvore geradora e o número de folhas dessa árvore fornece um limite inferior válido para nosso problema. Além disso, aplicamos a busca local sugerida em [43] a toda solução viável com valor maior que o melhor limite inferior disponível.

4.3 Formulação Direcionada

Na literatura, é comum encontrar problemas definidos sobre grafos não-orientados que são reformulados, com ganho, através de grafos direcionados. Nesse sentido, confira, por exemplo, a formulação de *Cut Sets* proposta para o Problema de Steiner em Grafos por ANEJA [50] e sua versão direcionada utilizada em [51, 22], dentre outros.

Seja $D = (V, A)$ um grafo direcionado, definido a partir de G da seguinte maneira: para cada aresta $e = (i, j) \in E$, dois arcos (i, j) e (j, i) são definidos em A . Adicionalmente, assuma que um vértice r , qualquer, é escolhido em V para atuar como raiz da arborescência que se quer construir. Assim sendo, para o vértice raiz, só adicionamos a A os arcos (r, j) e nenhum arco (j, r) , já que nenhum arco apontando para a raiz irá existir na arborescência. Com a transformação, algumas novas definições são necessárias, seja $\delta^+(i) \subseteq A$, $i \in V$, o conjunto de arcos (i, j) que saem de i e $\delta^-(i) \subseteq A$, $i \in V$, o conjunto

de arcos (j, i) que chegam em i . Introduzindo-se variáveis $y = \{y_a \in \mathbb{R}_+ : a \in A\}$ e utilizando-se, mais uma vez, as variáveis $z = \{z_i \in \{0, 1\} : i \in V\}$, como definidas anteriormente. Uma reformulação de (4.1)-(4.6) é, então, dada por

$$\max \sum_{i \in V} z_i \quad (4.18)$$

$$\text{s.a. } \sum_{a \in A} y_a = |V| - 1 \quad (4.19)$$

$$\sum_{a \in \delta^-(j)} y_a = 1 \quad j \in V \setminus \{r\} \quad (4.20)$$

$$\sum_{a \in A(S)} y_a \leq |S| - 1 \quad S \subset V, |S| \geq 2 \quad (4.21)$$

$$\sum_{a \in \delta^-(i)} y_a + \sum_{a \in \delta^+(i)} y_a + (|\delta^+(i)| - 1)z_i \leq |\delta^+(i)| \quad i \in V \setminus \{r\} \quad (4.22)$$

$$\sum_{a \in \delta^+(r)} y_a + (|\delta^+(r)| - 1)z_r \leq |\delta^+(r)| \quad (4.23)$$

$$0 \leq y_a \leq 1 \quad a \in A \quad (4.24)$$

$$z_i \in \{0, 1\} \quad i \in V. \quad (4.25)$$

Da mesma forma que anteriormente, $A(S)$ são todos os arcos $(i, j) \in A$, onde $i \in S$ e $j \in S$. A reformulação do MLSTP descrita acima resulta da substituição, em (4.2)-(4.6), das variáveis x_e , $e = (i, j) \in E$, por $(y_{ij} + y_{ji})$. Assim sendo, (4.19)-(4.21) e (4.24) corresponde a uma descrição alternativa do politopo da árvore geradora (veja [52] para mais detalhes). A formulação, imposta por (4.20), garante que exatamente um único arco deve existir na arborescência de solução apontando para cada vértice $i \in V \setminus \{r\}$. Com isso, a desigualdade (4.19) fica redundante. As desigualdades (4.22) e (4.23) correspondem à desigualdade (4.4) na formulação não direcionada.

Ao tentar resolver uma dada instância de MLSTP, em princípio, não teríamos vantagem alguma em escolher (4.18)-(4.25), em detrimento de (4.1)-

(4.6). Por outro lado, é fácil verificar que as desigualdades

$$y_a + z_i \leq 1 \quad a = (i, j) \in A, i \in V \setminus \{r\}, \quad (4.26)$$

não pode ser reproduzida em (4.1)-(4.6), obtemos uma formulação direcionada que domina a formulação não direcionada que lhe deu origem.

Podemos, ainda, ressaltar que as desigualdades (4.8) podem ser adaptadas para a nossa reformulação direcionada como

$$\sum_{a \in F^+} y_a + \sum_{a=(j,i) \in A | (i,j) \in F^+} y_a + (|F^+| - 1)z_i \leq |F^+| \quad i \in V, F^+ \subseteq \delta^+(i), |F^+| \geq 2. \quad (4.27)$$

Vale, também, notar que as desigualdades

$$\sum_{a \in \delta^+(i)} y_a + z_i \geq 1 \quad i \in V \setminus \{r\} \quad (4.28)$$

e

$$\sum_{a \in \delta^+(r)} y_a + z_r \geq 2 \quad (4.29)$$

são válidas para (4.18)-(4.25), pois se derivam diretamente de (4.7).

4.4 Formulação Multifluxos

Lançando mão de uma terceira descrição para o politopo da árvore geradora, proposta em [53] e discutida em [52], é possível obter outra reformulação de (4.1)-(4.6), tão forte quanto (4.18)-(4.25). Essa nova reformulação, no entanto, envolve apenas um número polinomial de restrições, o que permite resolver o MLSTP sem a necessidade de implementar algoritmos de solução específicos.

Seja $D = (V, A)$, definido a partir de G , como explicado anteriormente, e uma raiz $r \in V$, pré-especificada. Da raiz r , deverão ser enviados uma

unidade específica de mercadoria, para cada vértice em $v \in V \setminus \{r\}$. Essa operação é modelada através das variáveis de fluxo $\{f_a^k \in \mathbb{R}_+ : a = (i, j) \in A, k \in V \setminus \{r\}, i \neq k\}$. Dessa maneira, se um arco $a \in A$ é utilizado para enviar uma unidade da mercadoria k ao seu destino $k \in V \setminus \{k\}$, a variável f_a^k deverá assumir um valor igual a 1. Caso contrário, a variável deverá assumir um valor igual a 0. Como feito anteriormente, as variáveis $y = \{y_a \in \mathbb{R}_+ : a \in A\}$ são utilizadas para caracterizar os arcos de definição da arborescência que se quer construir ($y_a = 1$, se o arco faz parte da solução; $y_a = 0$, em caso contrário). Da mesma forma, as variáveis $z = \{z_i \in \{0, 1\} : i \in V\}$ são utilizadas para modelar as folhas da árvore geradora ($z_i = 1$ se o vértice $i \in V$ induz uma folha, e $z = 0$, em caso contrário). Uma outra reformulação de (4.1)-(4.6) é, então, dada por

$$\max \sum_{i \in V} z_i \quad (4.30)$$

$$\text{s.a. } \sum_{a \in A} y_a = |V| - 1 \quad (4.31)$$

$$y_a + z_i \leq 1 \quad a = (i, j) \in A, i \in V \setminus \{r\} \quad (4.32)$$

$$\sum_{a \in \delta^+(r)} y_a + (|\delta^+(r)| - 1)z_r \leq |\delta^+(r)| \quad (4.33)$$

$$\sum_{a \in \delta^+(r)} f_a^k = 1 \quad k \in V \setminus \{r\} \quad (4.34)$$

$$\sum_{a \in \delta^-(i)} f_a^k - \sum_{a \in \delta^+(i)} f_a^k = 0 \quad i, k \in V \setminus \{r\}; i \neq k \quad (4.35)$$

$$\sum_{a \in \delta^-(k)} f_a^k = 1 \quad k \in V \setminus \{r\} \quad (4.36)$$

$$f_a^k \leq y_a \quad a \in A, k \in V \setminus \{r\} \quad (4.37)$$

$$0 \leq f_a^k \leq 1 \quad a \in A, k \in V \setminus \{r\} \quad (4.38)$$

$$0 \leq y_a \leq 1 \quad a \in A \quad (4.39)$$

$$z_i \in \{0, 1\} \quad i \in V. \quad (4.40)$$

Aqui vale notar que as desigualdades (4.27), (4.28) e (4.29) são válidas para essa formulação.

4.5 Transformação para o PSG

A formulação proposta aqui é oriunda da transformação do MLSTP em PSG, como feito nas seções 2.2 e 3.2. Para transformar o problema, devemos primeiro definir como construir o grafo em dois níveis, a partir do grafo D . Seja $D = (V, A)$ um grafo direcionado, definido a partir de G da seguinte maneira: para cada aresta $(i, j) \in E$, dois arcos (i, j) e (j, i) são definidos em A . Considere o grafo em níveis $G_F = (V_F, A_F)$, onde o conjunto de vértices V_F é definido como

$$V_F = \{0\} \cup \{(i, h) : 1 \leq h \leq 2, i \in V\}$$

e o conjunto de arcos é definido como

$$\begin{aligned} A_F = & \{(0, (j, 1)) : j \in V\} \\ & \cup \{((i, 1), (j, 1)) : (i, j) \in A\} \\ & \cup \{((i, 1), (i, 2)) : i \in V\} \\ & \cup \{((i, 1), (j, 2)) : (i, j) \in A\}. \end{aligned}$$

Copiamos os vértices e as arestas de G duas vezes. Os vértices $(i, 1) \in V$ representarão o centro da árvore geradora, ou seja, os vértices não folhas ($\delta(i) \geq 2$). Já os vértices $(i, 2) \in V$ serão os vértices folhas da árvore geradora. Podemos ver que nenhum arco sai dos vértices $(i, 2)$. A adição do vértice 0 e dos arcos $(0, (j, 1))$, $j \in V$, são para ajudar na construção da árvore de Steiner, o vértice 0 é a raiz, e somente um arco $(0, (j, 1))$ estará na solução. Identificando-se de antemão um vértice $i \in V$ que não pode ser

folha, não será necessário adicionar o vértice 0 e os seus arcos adjacentes $(0, (j, 1))$. Já os arcos $((i, 1), (i, 2))$ são adicionados para garantir que um vértice i não folha tenha ligação a seu vértice terminal $(i, 2)$ correspondente. Na figura 4.3 é apresentado um exemplo da transformação do grafo original (figura 4.1) para o grafo em níveis (figura 4.2).

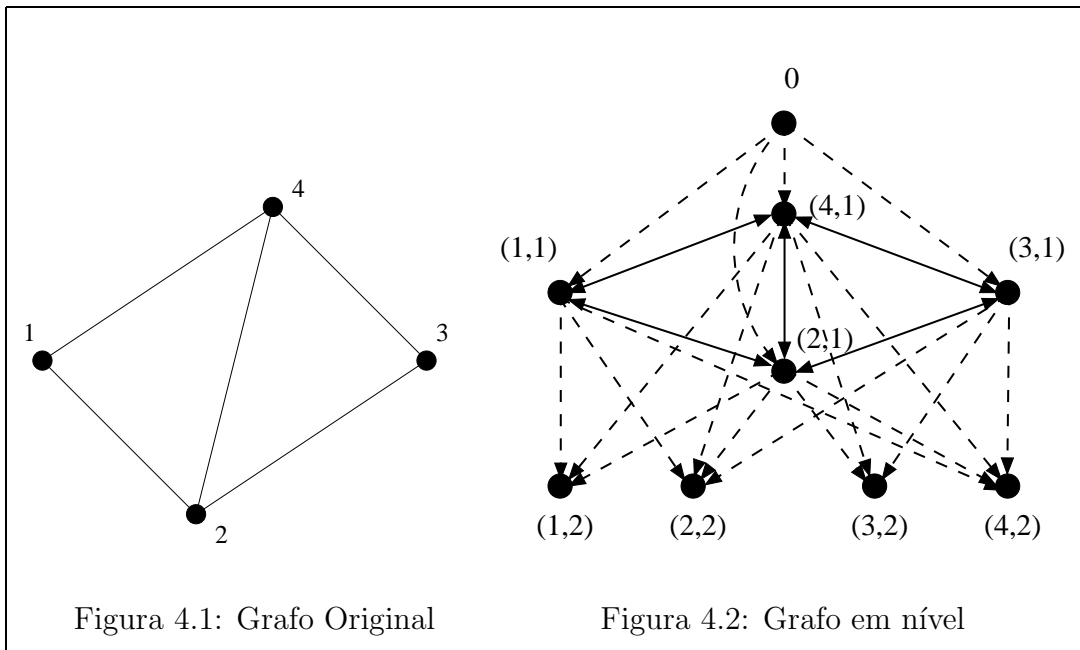


Figura 4.3: Exemplo da transformação do MLSTP para o PSG

Como feito para o HMSTP e o DMSTP, relacionaremos o MLSTP com uma árvore de Steiner no grafo G_F . A transformação direta para o PSG seria maximizar o número de arcos $((i, 1), (j, 2)), i \neq j$, com uma única restrição: ser uma árvore de Steiner direcionada com raiz em 0 e vértices terminais $R = \{(i, 2) : i \in V\}$. Entretanto, existe aqui uma diferença fundamental: os problemas anteriores eram de minimização enquanto o MLSTP é de maximização. Na aplicação de alguns modelos para o PSG, especificamente quando a topologia da solução desejada é a de uma árvore, é recomendado (em algumas formulações é até exigido) que o problema tratado seja o de

minimização sob custos positivos, ou, alternativamente, o de maximização sob custos negativos. Essa limitação pode ser facilmente contornada. No MLSTP, a função objetivo contém somente as variáveis referente às folhas; na transformação utilizada, essas variáveis correspondem aos arcos $((i, 1), (j, 2))$. Bastaria então impor aos demais arcos um custo suficientemente elevado, superior à soma dos custos de todos os arcos $((i, 1), (j, 2))$ juntos. Feito isso, o problema recairia então no modelo tradicional, ou seja, minimização sob custos não negativos. Como sugerido anteriormente para o DMSTP, os arcos $(0, (j, 1))$, $j \in V$, terão um custo ainda maior que os demais, para garantir que um único arco desse tipo apareça na árvore de Steiner.

Associando uma variável binária X_{0j} a cada arco $(0, (j, h))$ em G_F , uma variável binária X_{ij}^1 a cada arco $((i, 1), (j, 1))$ em G_F , uma variável binária X_{ij}^2 a cada arco $((i, 1), (j, 2))$ em G_F e associando uma variável binária X_{jj} para cada arco $((j, 1), (j, 2))$ em G_F , podemos usar qualquer modelo para o problema de Steiner no grafo em níveis, para fornecer um modelo válido para o MLSTP. Em seguida, apresentaremos um modelo.

4.5.1 Formulação por cortes orientados

Para descrever os cortes orientados, usaremos a mesma notação apresentada na seção 2.2.1. Temos a seguinte formulação:

$$\min \mathcal{C}^+ \sum_{j \in V} X_{0j} + \mathcal{C} \sum_{(i,j) \in A} X_{ij}^1 + \sum_{(i,j) \in A} X_{ij}^2 \quad (4.41)$$

$$\text{s.a.} \sum_{i \in V} X_{ij}^2 + X_{jj} = 1 \quad j \in V \quad (4.42)$$

$$X(\delta(S)) \geq 1 \quad S \in \mathcal{S} \quad (4.43)$$

$$X_{0j} \in \{0, 1\} \quad j \in V \quad (4.44)$$

$$X_{ij}^1 \in \{0, 1\} \quad (i, j) \in A \quad (4.45)$$

$$X_{ij}^2 \in \{0, 1\} \quad (i, j) \in A \quad (4.46)$$

$$X_{jj} \in \{0, 1\} \quad j \in V. \quad (4.47)$$

As restrições (4.42) garantem que cada vértice terminal seja visitado uma única vez. As restrições (4.43) são restrições de cortes orientados e garantem que a solução do problema possua pelo menos um arco em $\delta(S)$, o que torna a solução uma árvore de Steiner. A formulação descrita contém um número exponencial de restrições (4.43). O algoritmo de separação dessas restrições é o mesmo apresentado na seção 2.2.1.

4.6 Acelerando a resolução dos modelos

Nesta seção, apresentamos os métodos usados para acelerar a resolução das formulações orientadas e por cortes orientados.

4.6.1 Modelo por cortes orientados

O procedimento de resolução utilizado é o mesmo apresentado na seção 2.3. Primeiro é aplicada a heurística dual *Dual Ascent*, em seguida, a heurística primal. De posse de uma solução dual e de outra primal, tentamos então fixar variáveis pelo custo reduzido. Quando não é possível provar a otimalidade da melhor solução heurística encontrada, partimos então para a resolução do modelo exato, com a adição dos cortes identificados na heurística dual. Especificamente para o MLSTP, utilizamos apenas a heurística primal construtiva e as buscas locais apresentadas na seção 2.3.3.

4.6.2 Heurística primal para o MLSTP - Fujie

Utilizamos o método de resolução da relaxação de um subproblema do algoritmo *Branch-and-Bound* proposto por FUJIE [47], como uma heurística primal inicial. A relaxação de um subproblema, (4.15)-(4.17), pode ser resolvida através de um algoritmo para obter uma árvore geradora mínima, sob o custo das arestas modificadas. Para o subproblema raiz ($SP(\emptyset, \emptyset, V)$), um algoritmo de árvore geradora mínima estará implicitamente escolhendo as arestas (i, j) que possuem i e j com os maiores graus. Com isso, o algoritmo irá construir uma árvore geradora, onde os vértices de maior grau tenderão a ser vértices intermediários e aqueles de menor grau tenderão a ser folhas.

Essa heurística é usada no início do algoritmo *Branch-and-Cut* proposto para resolver a reformulação direcionada, apresentada na seção 4.3. É bom lembrar que uma das mudanças que fizemos no algoritmo *Branch-and-Bound* proposto por FUJIE [47] foi aproveitar a resolução da relaxação do subproblema para computar um limite inferior.

Ao longo da aplicação do algoritmo *Branch-and-Cut* para a resolução da formulação direcionada (4.18)-(4.25), também utilizamos a resolução da relaxação de um subproblema (4.15)-(4.17) como heurística. Na verdade, utilizamos a relaxação linear da formulação direcionada para modificar os custos das arestas. Isso é feito da seguinte forma: $c(i, j) = \bar{z}_i(\mathcal{W}+1)/(|\delta(i)|-1) + \bar{z}_j(\mathcal{W}+1)/(|\delta(j)|-1)$, onde \bar{z}_i é o valor da variável z_i na relaxação linear corrente e \mathcal{W} é um valor grande. Esse valor \mathcal{W} é uma alternativa para resolver a relaxação do subproblema onde o conjunto S_1 não é vazio. S_1 é o conjunto de vértices que são fixados para ser folha. Se o valor \mathcal{W} for grande o suficiente, uma aresta (i, j) , onde $i \in S_1$ ou $j \in S_1$, não será usada. Então, se $\bar{z}_i = 0$, tudo se passaria como se i pertencesse a S_0 e, de forma análoga, se $\bar{z}_i = 1$, como se i pertencesse a S_1 . Esta heurística é utilizada em todas

as iterações do nó raiz do algoritmo *Branch-and-Cut* e depois de resolver a relaxação linear nos outros nós.

4.6.3 Heurística primal para o MLSTP - Cobertura

Em quase todas as heurísticas propostas até o momento para o MLSTP, procura-se escolher os vértices com maiores graus em G para representar vértices intermediários da árvore geradora. Essa idéia pode ser refinada, levando-se em conta as interseções entre os vértices. Não é vantajoso escolher dois vértices de grau elevado para atuarem simultaneamente como vértices intermediários, se ambos têm o mesmo conjunto de vértices adjacentes.

A heurística de Cobertura, aqui proposta, mantém duas listas: uma contendo vértices intermediários e a outra vértices cobertos. A heurística é inicializada escolhendo-se um vértice para atuar como “raiz”. Em seguida, adicionamos o vértice raiz e os vértices adjacentes a ele, na lista dos vértices cobertos. Escolhemos então, na lista dos vértices cobertos, aquele de maior grau, reiniciamos a lista de cobertos e adicionamos à lista dos vértices intermediários o vértice escolhido. Também adicionamos os vértices adjacentes a esse vértice escolhido à lista de cobertos, exceto os vértices que já estão na mesma. Nas iterações seguintes, o processo utilizado é o basicamente o mesmo. Entretanto, na escolha de um vértice de maior grau na lista de vértices cobertos, só contabilizamos, para o grau de um dado vértice, as arestas onde a outra extremidade não faz parte de nenhuma das duas listas consideradas. A heurística é finalizada quando todos os vértices do grafo estiverem em uma dessas listas.

A heurística descrita acima é usada no início do algoritmo *Branch-and-Cut* proposto para resolver a reformulação direcionada, apresentada na seção 4.3. Como nessa formulação já escolhemos, de antemão, um vértice para

atuar como raiz, o mesmo vértice cumprirá o mesmo papel na heurística.

Durante a aplicação do algoritmo *Branch-and-Cut* para a formulação direcionada (4.18)-(4.25), também utilizamos a heurística de Cobertura para tentar encontrar novas soluções viáveis para o problema. Modificamos um pouco a heurística, para utilizar a relaxação linear da formulação direcionada como guia. Dessa maneira, são passados para a heurística os valores correntes, na relaxação linear, das variáveis z_i , $i \in V$. Na escolha de um vértice da lista de vértices cobertos para inserção na lista dos vértices intermediários, utilizamos como eventual critério de desempate, para vértices de mesmo grau, o valor de suas respectivas variáveis, na relaxação linear corrente. Esta heurística é utilizada em todas as iterações do nó raiz do algoritmo *Branch-and-Cut* e nos demais nós da árvore de procura, apenas quando resolvemos a relaxação linear.

4.6.4 Pós-processamento e busca local

Ao obtermos uma nova solução viável, pelas heurísticas de Fujie ou de Cobertura, aplicamos a busca local proposta em [43] apenas quando a nova solução tiver um valor superior a qualquer outra anteriormente obtida. Essa busca local consiste em trocar (uma ou duas) arestas da solução por outras (uma ou duas) arestas, de forma a melhorar a solução corrente. Além dessa busca local, aplicamos também um pós-processamento, descrito a seguir.

O pós-processamento consiste em percorrer todos os vértices intermediários de uma solução e verificar, para cada um deles, se é possível, com uma única excessão, conectar diretamente a outros vértices intermediários os vértices adjacentes ao vértice intermediário que estamos verificando. Em caso de êxito, uma solução viável, com uma folha a mais, terá sido encontrada. Descrevemos a seguir, o procedimento para um vértice intermediário i , numa solução

orientada. O procedimento, entretanto, é geral e independe da orientação da solução. No caso de uma solução orientada, os arcos que tentaremos remover são aquelas que saem de i . O único arco apontando para i deverá ser mantido, já que é ele que possibilita a conexão da raiz a i . É possível verificar que existem diversas maneiras de se implementar o procedimento indicado, entretanto, escolhemos uma que agiliza o mesmo. Vale aqui ressaltar que o caminho por nós seguido pode eventualmente impedir uma melhora na solução. Isso se justifica por não explorarmos todas as alternativas de troca disponíveis. No que fizemos, rotulamos inicialmente todos os descendentes de i , na árvore. Nesse caso, um descendente de i é um vértice para o qual existe um caminho levando de i ao mesmo. Feito isso, verificamos, para todo vértice j adjacente a i , a menos do vértice de chegada a i , se existe um vértice $z \in V$, intermediário e não rotulado (não descendente de i), que possui, no grafo D , uma aresta (z, j) . Em caso afirmativo, passamos a investigar o próximo vértice adjacente a i . Caso não exista tal vértice, interrompemos o procedimento para o vértice i . Se o movimento de transferência de vértices for bem sucedido, para todos os vértices candidatos adjacentes a i , é possível então melhorar a solução investigada, em uma unidade.

É possível tornar o procedimento mais abrangente, mas isso o tornaria mais lento. Em nossa experiência, a característica principal desejada para o mesmo é ser rápido, pois ele é utilizado ao longo da execução do algoritmo *Branch-and-Cut*, para toda solução viável que melhore a solução corrente. Não podemos esquecer que, aplicado o pós-processamento, aplicamos a seguir a busca local proposta em [43].

4.6.5 Teste de pré-processamento

Todos os testes de pré-processamento apresentados aqui são relativos aos vértices. Não encontramos, na literatura, nenhum teste desse tipo, aplicado a arestas.

O único teste de pré-processamento que encontramos na literatura foi:

Teste 4.6.1 (Grau 1) *Um vértice $i \in V$ com grau um em G , é um vértice folha. Se $|V| > 2$, o vértice j adjacente a i é vértice intermediário.*

Esse teste é trivial, já que, por definição, um vértice é folha quando o seu grau é um. Se $|V| = 2$, os dois vértices são folhas, caso contrário, o vértice j se conecta diretamente a i , já que é o único passível de fazê-lo. Conecta-se também, necessariamente, a pelo menos um outro vértice de V .

Um outro teste trivial, mas não utilizado até esse trabalho, é:

Teste 4.6.2 (Ponte) *Um vértice $i \in V$, que quando removido de V torna o grafo G desconexo, é um vértice intermediário.*

Como i é um vértice ponte, ou seja, um vértice cuja exclusão particiona o grafo em pelo menos duas componentes conexas disjuntas, qualquer árvore geradora de G tem que ter, no mínimo, uma aresta ligando um vértice de cada uma dessas componentes ao vértice i . Então, em qualquer árvore geradora de G , o grau de i é necessariamente maior ou igual ao número de componentes conexas que resultam da sua exclusão.

O último teste que utilizamos é um teste de dominância de vizinhança. Na descrição desse teste, denotamos por $N(i)$ o conjunto de vértices adjacentes a i , ou seja, qualquer vértice $j \in V$ tal que $(i, j) \in E$.

Teste 4.6.3 (Dominado) *Dados dois vértices $i, j \in V$ tais que $N(i) \supseteq N(j)$, existe uma árvore geradora de G com um número máximo de folhas, contendo i como vértice folha.*

Para qualquer árvore geradora de G com número máximo de folhas, se a vizinhança de $i \in V$ é dominada por algum outro vértice, podemos construir uma outra árvore geradora onde i é um vértice folha. Seja T essa árvore ótima. Se i é um vértice folha, não é necessário fazer nada. Se i e j são vértices intermediários, então a árvore T não pode ser ótima. Para verificar essa condição, suponha que T é ótima e que i e j são vértices intermediários de T . Então, existe um vértice z adjacente a i que também é intermediário. Note que z deve necessariamente existir para que a solução seja conexa, já que, como $N(i) \supseteq N(j)$, então $(i, j) \notin E$. Podemos então trocar as arestas (i, w) , $(i, w) \in T$ e $w \neq z$, pelas arestas (j, w) . Com isso, a solução obtida contém uma folha a mais, já que assumimos ser i um vértice intermediário. Dessa maneira, T não pode ser ótima. Se, por outro lado, i é intermediário e j é uma folha, existe um vértice z adjacente a i que também é intermediário. Podemos então trocar as arestas (i, w) , $(i, w) \in T$ e $w \neq z$, pelas arestas (j, w) . Com isso, i torna-se um vértice folha e j um vértice intermediário, não alterando o valor da solução.

Os testes de pré-processamento são aplicados antes da resolução das formulações orientadas e por cortes orientados.

4.7 Resultados

Nesta seção, comparamos o algoritmo *Branch-and-Bound* proposto em [47], com as melhorias apresentadas na seção 4.2, com a reformulação orientada, apresentada na seção 4.3, e com a formulação do problema transformado em PSG, apresentada na seção 4.5.

O único estudo computacional que encontramos na literatura foi aquele apresentado em [47]. Nesse trabalho, as instâncias usadas foram obtidas

gerando-se arestas aleatórias até o grafo se tornar conexo e atingir uma certa densidade. Para cada densidade e número de vértices especificados, 10 instâncias foram geradas. A densidade imposta aos grafos variou de 10% a 70%, enquanto o número de vértices variou de 30 a 100. Entretanto, para as instâncias com mais de 60 vértices, são apresentados resultados em [47], apenas para instâncias de maior densidade.

As instâncias aqui utilizadas são construídas de forma diferente daquela sugerida em [47]. Primeiro, construímos um caminho hamiltoniano. Em seguida, adicionamos arestas de forma aleatória, até atingirmos a densidade desejada. Variamos a densidade de 5% a 70% e o número de vértices de 30 a 150.

Os resultados computacionais aqui apresentados foram obtidos em um PC Intel Core 2 Duo, 2.2 GHz com 2Gb de RAM, utilizando o programa XPRESS 2007A para resolver as relaxações lineares e a programação inteira, quando necessário.

Na tabela 4.1, apresentamos os resultados dos três algoritmos: *Branch-and-Bound* (B&B), reformulação orientada (Dir) e transformado em PSG (PSG). Nas colunas referentes ao algoritmo *Branch-and-Bound*, apresentamos o melhor resultado dos dois tipos de busca (*depth-first* e *best bound*). Nas três primeiras colunas, indicamos o número de vértices, a densidade e o valor da solução ótima, para cada instância considerada. Em seguida, indicamos para cada algoritmo o valor da relaxação linear, o número de nós necessários para provar otimalidade, o tempo computacional e a melhor solução inicial encontrada (cada algoritmo possui uma ou mais heurísticas iniciais). Os campos com “-” significam que não foi possível resolver a instância por causa de um dos motivos a seguir: excedeu 24 horas de processamento, problema relacionado a excesso de memória.

V	d	Ótimo	B&B				Dir				SPG			
			LP	Nós	T(s)	Heu	LP	Nós	T(s)	Heu	LP	Nós	T(s)	Heu
30	10	15	17.83	291	0.12	15	15.57	1	0.01	15	15.80	1	0.04	15
	20	23	26.04	5055	0.33	22	24.48	7	0.1	22	23.95	1	0.12	22
	30	26	27.44	842	0.24	24	27.05	1	0.03	25	26.13	5	26.7	25
	50	27	28.46	307	0.19	26	28.13	3	0.09	27	27.94	1	1.28	25
	70	28	28.83	1	0.16	28	28.73	1	0.01	28	28.00	1	0.26	26
50	5	19	21.50	265	0.94	19	19.00	1	0.02	19	19.00	1	0.09	16
	10	38	42.16	82599	4.54	34	39.75	41	0.82	36	38.86	38	94	35
	20	43	46.58	225771	16.9	42	45.22	77	1.32	43	44.48	57	1827	42
	30	45	47.59	38155	5.97	42	46.80	39	1.21	45	46.08	43	22424	43
	50	47	48.45	3050	4	46	48.18	13	0.51	47	47.36	-	-	44
	70	48	48.82	5	1.64	47	48.62	1	0.09	48	48.00	1	2.08	-
70	5	43	49.57	9068999	313	41	44.45	53	0.99	40	43.56	19	103	39
	10	57	63.18	-	-	52	59.20	174	4.73	56	58.60	-	-	53
	20	63	66.22	-	-	60	65.04	607	16.3	62	64.37	-	-	60
	30	65	67.62	4113677	536	62	66.91	35	2.9	64	66.15	-	-	62
	50	67	68.52	33058	25.3	65	68.14	7	1.33	67	-	-	-	65
	70	68	68.76	2661	10.8	67	68.68	5	1.92	67	68	1	8.55	66
100	5	76	83.42	-	-	66	79.36	605	24.5	69	78.57	-	-	69
	10	87	93.27	-	-	77	89.52	135	9.36	86	-	-	-	80
	20	92	96.56	-	-	86	94.85	1025	86.1	91	-	-	-	88
	30	94	97.39	-	-	92	96.68	1753	258	93	-	-	-	90
	50	96	98.36	348389	213	95	98.03	479	132	96	-	-	-	95
	70	97	98.76	9091	50.5	97	98.64	121	154	97	-	-	-	-
120	5	95	105.04	-	-	85	97.77	24	2.65	92	-	-	-	87
	10	107	113.16	-	-	97	109.83	869	65.4	103	-	-	-	98
	20	112	116.39	-	-	107	114.93	2401	393	112	-	-	-	108
	30	114	117.40	-	-	111	116.69	2301	653	114	-	-	-	112
	50	116	118.42	571335	435	115	118.12	1297	815	116	-	-	-	113
	70	117	118.72	13791	97	116	118.63	137	356	117	-	-	-	116
150	5	124	135.11	-	-	112	128.74	31077	2954	122	-	-	-	114
	10	136	142.81	-	-	125	139.59	6089	3247	134	-	-	-	128
	20	141	146.81	-	-	135	145.12	173425	61639	140	-	-	-	137
	30	144	147.38	-	-	140	146.67	3043	2617	143	-	-	-	140
	50	146	148.38	2104992	2190	144	148.10	1755	2756	146	-	-	-	144
	70	147	148.72	21625	301	147	148.63	219	1828	147	-	-	-	145
200	5	-	185.22	-	-	153	177.88	-	-	166	-	-	-	159
	10	-	193.21	-	-	172	189.70	-	-	182	-	-	-	178
	20	-	196.33	-	-	185	195.13	-	-	190	-	-	-	184
	30	-	197.35	-	-	189	196.77	-	-	193	-	-	-	190
	50	196	198.32	-	-	195	198.07	3125	20155	195	-	-	-	193
	70	197	198.73	38215	1322	196	198.63	253	8154	197	-	-	-	195

Tabela 4.1: Resultados dos algoritmos para o MLSTP

Como podemos notar na tabela 4.1, a melhor relaxação linear foi a da transformação para o PSG. Entretanto, o algoritmo de melhor desempenho foi o da reformulação orientada. Mesmo com as melhorias implementadas no algoritmo *Branch-and-Bound* proposto em [47], o comportamento daquele algoritmo foi o mesmo apresentado em [47]. Para instância de menor densidade, o algoritmo *Branch-and-Bound* tem um desempenho inferior. Possivelmente, esse foi o motivo para que, nos testes realizados em [47], para instâncias envolvendo de 60 a 100 vértices, apenas as de maior densidade terem sido testadas (para as instâncias de 70 vértices, densidades a partir de 30% e, para as instâncias de 100 vértices, densidades a partir de 50%). A reformulação direcionada obteve o melhor resultado para quase todas as instâncias (31 em 37). O algoritmo *Branch-and-Bound* se mostrou melhor para as maiores instâncias (maior densidade e maior número de vértices). Esse resultado é coerente, uma vez que, para essas instâncias, a solução ótima e a relaxação linear já estão muito próximas do limite máximo do problema, ou seja, $(|V| - 1)$. Com isso, a diferença entre os limites superiores cai. Podemos também notar que as heurísticas usadas no *Branch-and-Bound* melhoram o desempenho do mesmo, para as instâncias com maior densidade.

Na tabela 4.2, apresentamos os resultados das heurísticas de forma mais detalhada. Nas três primeiras colunas, indicamos o número de vértices, a densidade e o ótimo de cada instância. Nas próximas duas colunas, indicamos o valor da solução inicial encontrada com a heurística Fújie e Cobertura, apresentadas nas seções 4.6.2 e 4.6.3, respectivamente. Indicamos a melhor solução encontrada durante o *Branch-and-Cut* da reformulação orientada, na sexta coluna, e da formulação da transformação em PSG, na sétima coluna. Nas cinco últimas colunas, indicamos o tempo total (heurísticas iniciais e durante o *Branch-and-Cut*) gasto nas heurísticas Fújie, Cobertura, para o

PSG com busca local (ver seção 2.3.4), pós-processamento (ver seção 4.6.4) e busca local (proposta em [43]).

V	d	Ótimo	Inicial		B&C		T_FJ(s)	T_Cb(s)	T_PSG(s)	T_Pos()	T_BL()
			Fujie	Cobertura	Dir	SPG					
30	10	15	12	15	-	0	0,00	0,00	0,00	0,00	0,00
	20	23	18	22	23	23	0,00	0,00	0,00	0,00	0,00
	30	26	22	25	26	0	0,00	0,00	0,01	0,00	0,00
	50	27	24	27	-	27	0,00	0,00	0,00	0,00	0,00
	70	28	28	28	-	28	0,00	0,00	0,00	0,00	0,00
50	5	19	13	19	-	19	0,00	0,00	0,01	0,00	0,01
	10	38	29	36	38	36	0,00	0,00	0,00	0,00	0,01
	20	43	39	43	-	43	0,00	0,00	0,05	0,00	0,00
	30	45	42	45	-	45	0,00	0,00	0,16	0,00	0,00
	50	47	46	47	-	46	0,00	0,00	-	0,00	0,00
70	48	47	48	-	48	0,00	0,00	0,01	0,00	0,00	
70	5	43	33	40	43	40	0,00	0,00	0,03	0,03	0,05
	10	57	51	56	57	0	0,03	0,00	-	0,00	0,02
	20	63	57	62	63	0	0,02	0,01	-	0,00	0,03
	30	65	61	64	65	63	0,00	0,00	-	0,00	0,03
	50	67	63	67	-	66	0,00	0,00	-	0,00	0,00
70	68	66	67	68	68	0,00	0,00	0,00	0,00	0,05	
100	5	76	59	69	76	0	0,08	0,01	-	0,10	0,13
	10	87	70	86	87	83	0,00	0,00	-	0,00	0,01
	20	92	86	91	92	89	0,11	0,04	-	0,00	0,06
	30	94	90	93	94	93	0,14	0,04	-	0,00	0,19
	50	96	93	96	-	96	0,04	0,02	-	0,00	0,00
70	97	96	97	-	97	0,01	0,03	-	0,00	0,16	
120	5	95	70	92	95	88	0,01	0,00	-	0,00	0,17
	10	107	92	103	107	101	0,13	0,04	-	0,00	0,21
	20	112	102	112	-	109	0,30	0,11	-	0,00	0,00
	30	114	108	114	-	113	0,36	0,13	-	0,00	0,00
	50	116	115	116	-	115	0,20	0,06	-	0,00	0,00
70	117	114	117	-	117	0,05	0,00	-	0,00	0,00	
150	5	124	107	122	124	115	5,66	1,75	-	0,12	0,08
	10	136	119	134	136	131	1,64	0,63	-	0,00	0,16
	20	141	133	140	141	138	86,44	20,79	-	0,00	0,51
	30	144	139	143	144	142	0,74	0,22	-	0,00	0,44
	50	146	141	146	-	145	0,49	0,13	-	0,00	0,00
70	147	145	147	-	147	0,05	0,00	-	0,00	0,00	
200	5	-	141	166	172	162	0,00	0,00	-	-	-
	10	-	166	182	184	180	0,00	0,00	-	-	-
	20	-	181	190	191	188	0,00	0,00	-	-	-
	30	-	186	193	-	192	0,00	0,00	-	-	-
	50	196	193	195	196	194	1,56	0,38	-	0,00	2,29
70	197	195	197	-	196	0,17	0,05	-	0,00	0,00	

Tabela 4.2: Resultados das heurísticas para o MLSTP

Mesmo sem ser o objetivo principal desta tese, os resultados da heurística proposta (Cobertura) apresentaram um bom desempenho, soluções com valores melhores e um tempo computacional menor que os da alternativa testada. O resultado foi tão bom, que a busca local e o pós-processamento quase não

melhoraram as soluções encontradas.

4.8 Comentários Finais

Apresentamos três formulações para o MLSTP. Dessas três formulações, implementamos dois algoritmos *Branch-and-Cut* para resolver duas delas. Um desses algoritmos é o mesmo utilizado para o HMSTP e DMSTP. Para o outro algoritmo foram adicionadas duas heurísticas, uma busca local e um pós-processamento.

Os resultados dos algoritmos foram superiores aos existentes na literatura, mesmo após refinarmos o algoritmo proposto em [47]. O algoritmo envolvendo a reformulação para o PSG obteve a menor diferença entre a relaxação linear e a solução ótima. Entretanto, o tempo requerido para resolver a relaxação linear correspondente, é excessivo. Já o algoritmo da reformulação orientada obteve o melhor desempenho, conseguindo resolver, com garantia de otimalidade, um número maior de instâncias, em menor tempo computacional.

Capítulo 5

Conclusões

Nesta tese, apresentamos três trabalhos que tratam de problemas que envolvem a determinação de árvores geradoras ótimas em grafos.

Na primeira parte, apresentamos o problema de árvore geradora mínima com restrição de nível, construímos um grafo em nível a partir do grafo original e propusemos três formulações usando esse grafo em nível. Na verdade, podemos ver o problema nesse novo grafo em nível como um problema de Steiner em grafo direcionado. Vários métodos foram incorporados ao algoritmo de resolução do modelo proposto, objetivando melhorar o desempenho do algoritmo. Esses métodos, exceto um, foram tirados da literatura do problema de Steiner em grafos. Apresentamos, ainda, uma nova heurística com busca local para o problema tratado, ou seja, o HMSTP. Mesmo sendo uma heurística simples, ela cumpriu o papel de construir soluções, de qualidade razoável, de forma muito rápida. Os resultados computacionais obtidos em nosso estudo nos pareceu expressivos, pois conseguimos resolver, de forma exata, instâncias teste do problema em tempos de CPU muito menores que aqueles reportados na literatura. Da mesma forma, resolvemos instâncias do HMSTP com o dobro do tamanho da maior instância resolvida na literatura,

provando a otimalidade de soluções para instâncias com até 160 vértices.

Na segunda parte, investigamos o problema da árvore geradora mínima com restrição de diâmetro, ou seja, o DMSTP. Para esse problema, apresentamos uma reformulação para o problema de Steiner direcionado, como feito na primeira parte da tese para o HMSTP. O algoritmo de solução proposto para o DMSTP foi praticamente o mesmo, com alguns ajustes. Em termos relativos, os resultados computacionais aqui obtidos foram ainda melhores que aqueles obtidos na primeira parte da tese. Com o nosso trabalho, o tempo de CPU necessário para resolver instâncias teste do problema caiu drasticamente. Em relação a dimensão das instâncias resolvidas com garantia de otimalidade, o ganho foi ainda maior. Antes, só eram resolvidas instâncias de até 25 vértices, em grafos completos, e 60 vértices, em grafos esparsos. Conseguimos resolver instâncias com até 160 vértices, para grafos completos.

Na terceira e última parte, apresentamos o problema de árvore geradora com um número máximo de folhas, ou seja, o MLSTP. Apresentamos uma reformulação mais forte que a formulação que lhe deu origem, na literatura. Também apresentamos uma reformulação do MLSTP como um problema de Steiner em grafos. Uma nova heurística foi proposta para o problema com um procedimento de pós-processamento. Os resultados obtidos com a primeira reformulação dominam aqueles encontrados na literatura. Já a segunda reformulação levou a relaxações lineares mais fortes que aquelas obtidas por qualquer outra formulação proposta para o problema. Isso ocorre, entretanto, a um elevado tempo de CPU. Dessa forma, o desempenho do algoritmo proposto para nossa segunda reformulação do MLSTP é dominado por aquele proposto para a nossa primeira reformulação do problema.

nova transformação também apresentou uma diferença entre a relaxação

linear e a solução ótima menor que as outras formulações, mas a um custo computacional muito grande. Com isso, a reformulação obteve um resultado computacional melhor.

Referências Bibliográficas

- [1] PRIM, R. C. “Shortest Connection Networks and some Generalizations”. *Bell System Technical Journal*, v. 36, n. 6, pp. 1389–1401, 1957.
- [2] KARP, R. *Reducibility Among Combinatorial Problems*, pp. 85–103. Complexity of Computer Computations. Plenum Press, New York, 1972.
- [3] HWANG, F.K., RICHARDS, D.S., WINTER, P. *The Steiner Tree Problems*, v. 53, *Annals of Discrete Mathematics*. North-Holland, 1992.
- [4] MACULAN, N. “THE Steiner Problem in Graphs”. *Annals of Discrete Mathematics*, v. 31, pp. 185–212, 1987.
- [5] LUCENA, A., BEASLEY, J. “A Branch and Cut Algorithm for the Steiner Problem in Graphs”. *Networks*, v. 31, pp. 39–59, 1998.
- [6] UCHOA, E. *Algoritmos para Problemas de Steiner com Aplicações em Projeto de Circuitos VLSI*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro, 2001.
- [7] DU, D., SMITH, J.M., RUBINSTEIN, J.H. (eds). *Advances in Steiner Trees*, v. 6, *Combinatorial Optimization*. Springer, 2000.
- [8] WOLSEY, L. A. *Integer Programming*. Wiley & Sons, New York, 1998.

- [9] GOUVEIA, L. “Using the Miller-Tucker-Zemlin Constraints to Formulate a Minimal Spanning Tree Problem with Hop Constraints”. *Computers & Operations Research*, v. 22, pp. 959–970, 1995.
- [10] DAHL, G. “The 2-Hop Spanning Tree Problem”. *Operations Research Letters*, v. 23, pp. 21–26, 1998.
- [11] MANYEM, P., STALLMANN, M. *Some Approximation Results in Multicasting*. North Carolina State University, 1996.
- [12] WOOLSTON, K., ALBIN, S. “The Design of Centralized Network with Reliability and Availability Constraints”. *Computers & Operations Research*, v. 15, pp. 207–217, 1988.
- [13] GOUVEIA, L. “Multicommodity Flow Models for Spanning Trees with Hop Constraints”. *European Journal of Operational Research*, v. 95, pp. 178–190, 1996.
- [14] GOUVEIA, L. “Using Variable Redefinition for Computing Lower Bounds for Minimum Spanning and Steiner Trees with Hop Constraints”. *INFORMS Journal on Computing*, v. 10, pp. 180–188, 1998.
- [15] GOUVEIA, L., REQUEJO, C. “A New Lagrangian Relaxation Approach for the Hop-Constrained Minimum Spanning Tree Problem”. *European Journal of Operational Research*, v. 132, pp. 539–552, 2001.
- [16] DAHL, G., FLATBERT, T., FOLDNES, N., GOUVEIA, L. *The Jump Formulation for the Hop-Constrained Minimum Spanning Tree Problem*. Technical report, Centro de Investigação Operacional, Faculdade de Ciências da Universidade de Lisboa, 2004.

- [17] DAHL, G., GOUVEIA, L., REQUEJO, C. *On Formulations and Methods for the Hop-Constrained Minimum Spanning Tree Problem*, pp. 493–515. Handbooks of Telecommunications. Springer, 2006.
- [18] WONG, R. “A Dual Ascent Approach for Steiner Tree Problems on a Directed Graph”. *Mathematical Programming*, v. 28, pp. 271–287, 1984.
- [19] CLAUS, A., MACULAN, N. *Une Nouvelle Formulation du Problème de Steiner Sur un Graphe*. Technical Report 280, Centre de Recherche sur les Transports, Université de Montréal, 1983.
- [20] AHUJA, R., MAGNANTI, T., ORLIN, J. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, New Jersey, 1993.
- [21] HAO, J., ORLIN, J. B. “A Faster Algorithm for Finding the Minimum Cut of a Graph”. *Proc. 3rd ACM-SIAM Symposium on Discrete Algorithms*, pp. 165–174, 1992.
- [22] KOCH, T., MARTIN, A. “Solving Steiner tree problems in graph to optimality”. *Networks*, v. 32, pp. 207–232, 1998.
- [23] SKOROBOHATYJ, G. “Mathprog: Codes for Finding Minimum Cuts of Several Kinds in Directed and Undirected Graphs”. <http://elib.zib.de/pub/Packages/mathprog/mincut/index.html>.
- [24] TAKAHASHI, H., MATSUYAMA, A. “An Approximate Solution for the Steiner Problem in Graphs”. *Mathematica Japonica*, v. 24, pp. 573–577, 1980.
- [25] MINOUX, M. “Efficient greedy heuristics for Steiner tree problems using reoptimization and supermodularity”. *INFORMS Journal on Computing*, v. 28, pp. 221–233, 1990.

- [26] VOSS, S. *Steiner-Probleme in Graphen*. Anton Hain, 1990.
- [27] VERHOEVEN, M. G. A., SEVERENS, M. E. M., AARTS, E. H. L. *Modern Heuristic Search Methods*, chapter Local search for Steiner tree problems in graphs, pp. 117–129. Wiley, 1996.
- [28] DUIN, C. W. *Steiner's Problem in Graphs*. PhD thesis, University of Amsterdam, 1993.
- [29] POLZIN, T., DANESHMAND, S. V. “Improved Algorithms for the Steiner Problem in Networks”. *Discrete Applied Mathematics*, v. 112, n. 1-3, pp. 263–300, 2001.
- [30] GAREY, M. R., JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.
- [31] GOUVEIA, L., MAGNANTI, T. L. *Modelling and Solving the Diameter-Constrained Minimum Spanning Tree Problem*. Technical report, DEIO-CIO, Faculdade de Ciências, 2000.
- [32] ACHUTHAN, N. R., CACCETTA, L., CACCETTA, P., GEELLEN, J. F. “Algorithms for the Minimum Weight Spanning Tree with Bounded Diameter Problem”. In PHUA, P. K. H., WANG, C. M., YEONG, W. Y., LEONG, T. Y., LOH, H. T., TAN, K. C., CHOU, F. S. (eds), *Optimization: Techniques and Applications*, v. 1, pp. 297–304. World Scientific Publishing, 1992.
- [33] ACHUTHAN, N. R., CACCETTA, L., CACCETTA, P., GEELLEN, J. F. “Computational Methods for the Diameter Restricted Minimum Weight Spanning Tree Problem”. *Australasian Journal of Combinatorics*, v. 10, pp. 51–71, 1994.

- [34] GOUVEIA, L., MAGNANTI, T. L. “Network Flow Models for Designing Diameter-Constrained Minimum-Spanning and Steiner Trees”. *Networks*, v. 41, pp. 159–173, 2003.
- [35] GOUVEIA, L., MAGNANTI, T. L., REQUEJO, C. “A 2-Path Approach for Odd-Diameter-Constrained Minimum Spanning and Steiner Trees”. *Networks*, v. 44, pp. 254–265, 2004.
- [36] SANTOS, A. C., LUCENA, A., RIBEIRO, C. C. “Solving Diameter Constrained Minimum Spanning Tree Problems in Dense Graphs”. In *Experimental and Efficient Algorithms*, v. 3059, *Lecture Notes in Computer Science*, pp. 458–467. Springer, Berlin, 2004.
- [37] GRUBER, M., RAIDL, G. R. “A New 0-1 ILP Approach for the Bounded Diameter Minimum Spanning Tree Problem”. In GOUVEIA, L., MOURÃO, C. (eds), *Proceedings of the 2nd International Network Optimization Conference*, v. 1, Portugal, 2005.
- [38] NORONHA, T. F., C.SANTOS, A., RIBEIRO, C. C. “Constraint programming for the diameter constrained minimum spanning tree problem”. In *Proceedings of IV Latin-American Algorithms, Graphs and Optimization Symposium*, *Electronic Notes in Discrete Mathematics*, Puerto Varas, Chile, 2008. To appear.
- [39] GUHA, S., KHULLER, S. “Approximation algorithms for connected dominating sets”. In *Proceedings of the Fourth Annual European Symposium on Algorithms*, pp. 179–194, 1996.
- [40] STORER, J. A. “Constructing full Spanning Trees for Cubic Graphs”. *Information Processing Letters*, v. 13, pp. 8–11, 1981.

- [41] FERNANDES, L. M., GOUVEIA, L. “Minimal Spanning trees with a constraint on the number of leaves”. *European Journal of Operational Research*, v. 104, pp. 250–261, 1998.
- [42] LU, H., RAVI, R. “The power of local optimization: approximation algorithms for maximum-leaf spanning tree”. In *Thirtieth Annual Allerton Conference on Communication*, pp. 533–542, 1992.
- [43] LU, H., RAVI, R. “Approximating Maximum Leaf Spanning Trees in Almost Linear Time”. *Journal of Algorithms*, v. 29, pp. 132–141, 1998.
- [44] SOLIS-OBA, S. “2-approximation algorithm for finding a spanning tree with maximum number of leaves”. *Lecture notes in Computer Science*, v. 1461, pp. 441–452, 1998.
- [45] GALBIATI, G., MAFFIOLI, F., MORZENTI, A. “A short note on the approximability of the maximum leaves spanning tree problem”. *Information Processing Letters*, v. 52, pp. 45–49, 1994.
- [46] FUJIE, T. “The maximum-Leaf Spanning Tree Problem: Formulations and Facets”. *Networks*, v. 43, n. 4, pp. 212–223, 2004.
- [47] FUJIE, T. “An exact algorithm for the maximum-leaf spanning tree problem”. *Computers & Operations Research*, v. 30, pp. 1931–1944, 2003.
- [48] EDMONDS, J. “Matroids and the greedy algorithm”. *Mathematical Programming*, v. 1, pp. 127–136, 1911.
- [49] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., STEIN, C. *Introduction to Algorithms*. MIT Press, 2001.

- [50] ANEJA, Y. P. “An integer linear programming approach to the Steiner problem in graphs”. *Networks*, v. 10, pp. 167–178, 1980.
- [51] CHOPRA, S., GORRES, E. R., RAO, M. R. “Solving the Steiner tree problem on a graph using branch and cut”. *ORSA Journal on Computing*, v. 4, pp. 320–335, 1992.
- [52] MAGNANTI, T. L., WOLSEY, L. A. *Network Models*, v. 7, *Handbooks in Operations Research and Management Science*, chapter Optimal trees, pp. 503–615. North Holland, 1995.
- [53] MACULAN, N. “A new linear programming formulation for the shortest s-directed spanning tree problem”. *Journal of Combinatorics Information and Systems Science*, v. 31, pp. 53–56, 1986.