Árvores k-restritas e aproximações para o Problema de Steiner em Grafos

Eduardo Kazuaki Gondo

DISSERTAÇÃO APRESENTADA AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA DA
UNIVERSIDADE DE SÃO PAULO
PARA OBTENÇÃO DO GRAU DE
MESTRE EM
CIÊNCIA DA COMPUTAÇÃO

Orientadora: Prof^a. Dr^a. Cristina Gomes Fernandes

— São Paulo, março de 2002 —

Árvores k-restritas e
aproximações para o
Problema de Steiner em Grafos

Este exemplar corresponde à redação final da dissertação,
devidamente corrigida, defendida por
Eduardo Kazuaki Gondo e aprovada pela comissão julgadora.

São Paulo, 10 de maio de 2002.

Banca examinadora:

- Prof^a. Dr^a. Cristina Gomes Fernandes (orientadora) — IME-USP
- Prof. Dr. Carlos Eduardo Ferreira — IME-USP
- Prof. Dr. Flávio Keidi Miyazawa — IC-UNICAMP

Aos meus pais e
ao meu irmão,

Agradecimentos

Os principais objetivos deste trabalho foram aprender todos os passos da pesquisa científica ou, como gosto de dizer, "aprender a aprender" e conseguir expressar as idéias e resultados dessa pesquisa de forma clara e precisa. Porém os primeiros passos de qualquer aprendizado não é feito solitariamente, é preciso a ajuda de várias pessoas. Para essas pessoas vão meus sinceros agradecimentos.

À minha orientadora, Professora Cristina Gomes Fernandes, pela excelente orientação, pelos ensinamentos, constante presença, dedicação, atenção, paciência, estímulo e amizade.

Aos meus pais e ao meu irmão pelo apoio, estímulo e amor.

Ao meu amigo Carlos que me incentivou ao ingresso do mestrado e pela sua compreensão nesta "reta final".

Aos professores, Ivan de Camargo, José Augusto Soares e Stavros Christodoulou por acreditarem na realização deste mestrado.

Aos Professores Paulo Feofiloff e Carlos Eduardo Ferreira pelas suas dicas no início deste trabalho.

Aos meus amigos do Dieese pelo estímulo e companheirismo.

Aos meus amigos do IME pelos momentos de descontração.

À Simone pela compreensão, paciência e estímulo.

Resumo

O Problema de Steiner em Grafos consiste em encontrar, em um grafo com custos nas arestas, uma árvore de custo mínimo contendo um dado conjunto de vértices.

Esse problema tem um papel central na área de algoritmos de aproximação e diversas de suas variantes têm aplicações, por exemplo, em projeto de redes de comunicação.

Neste trabalho apresentamos um estudo de vários algoritmos de aproximação para este problema. Os algoritmos escolhidos baseiam-se na utilização de árvores k-restritas e formam uma série, cada um deles tendo explorado melhor uma idéia introduzida pelo anterior. Todos eles foram projetados durante a última década, sendo o mais recente, o melhor algoritmo de aproximação para o problema até o momento em termos de razão de aproximação.

Abstract

The Steiner Tree Problem consists in finding a minimum cost tree that spans a given subset of vertices. This problem plays a central role in approximation algorithms. Many of its variants have applications on network design.

In this work we present a study of several approximation algorithms for the Steiner Tree Problem. The selected algorithms are based on the use of k-restricted trees and form a series, each one exploring further an idea introduced by the previous one. All of them were developed during the last decade and the last of them has the best performance ratio known for the problem so far.

Sumário

Introdução

Problemas de otimização são de interesse nas mais diversas áreas do conhecimento. Chamamos de *Otimização Combinatória* a área da Ciência da Computação que estuda esses tipos de problemas. Um problema de otimização combinatória pode ser definido basicamente da seguinte forma: *dada uma função definida sobre um conjunto finito A, encontre, se existir, um elemento de A que minimize (maximize) a função dada.*

Embora a definição seja simples, para muitos desses problemas não é fácil encontrar um elemento de A que responda a pergunta do problema. Em geral, a dificuldade está no fato que A, mesmo sendo finito, pode ter um número muito grande de elementos. Isso torna inviável, do ponto de vista computacional, gerar e verificar todos os elementos de A para encontrar um melhor. Vários desses problemas fazem parte do conjunto dos problemas $\mathcal{NP}$*-difíceis*. Isso significa que, para vários desses problemas, não conhecemos e nem mesmo esperamos encontrar algoritmos eficientes para resolvê-los. Contudo não podemos simplesmente deixar de estudar tais problemas, porque eles possuem importantes aplicações práticas.

Cormen, Leiserson e Rivest [10] sugerem duas abordagens para problemas $\mathcal{NP}$-difíceis:

- Algoritmos que produzam soluções ótimas mas que no pior caso consumam tempo exponencial no tamanho da entrada. (Estes algoritmos podem ser adequados apenas para instâncias pequenas ou particulares.)

- Algoritmos eficientes (isto é, que consomem tempo polinomial no tamanho da entrada) que produzam *soluções aproximadas* com uma garantia de qualidade, ou seja, um limite para o erro entre o custo da solução produzida pelo algoritmo e o custo de uma solução ótima do problema. Estes algoritmos são denominados *algoritmos de aproximação* e chamamos *razão de aproximação* a sua garantia de qualidade. (A definição precisa desses conceitos será dada no capítulo 1.)

O *Problema de Steiner em Grafos* é um dos exemplos de problema de otimização $\mathcal{NP}$-difícil [18]. Informalmente, ele pode ser assim definido: *dado um grafo G com custos não-negativos nas arestas, determinar um subgrafo conexo de G de custo mínimo que contenha um dado conjunto de vértices.* Esse problema aparece no contexto de projetos de circuitos VLSI [29], árvores filogenéticas [28], problemas de conexão [23], etc.

O surgimento do *Problema de Steiner* data do século XVII. Segundo Maculan [35], a primeira formulação do problema foi feita pelo matemático francês Pierre de Fermat. O

problema era o seguinte: *dadas três cidades, deseja-se construir uma rede de estradas de comprimento total mínimo de forma que se possa ir de qualquer uma das três cidades a qualquer outra através dessas estradas.*

Utilizando geometria, podemos reescrever formalmente o problema acima: *dados três pontos no plano euclidiano, determinar um conjunto de segmentos de reta que interligue estes pontos de tal forma que a soma dos comprimentos dos segmentos de reta seja mínima.*

Ainda de acordo com Maculan, no século XIX, o geômetra alemão Jacob Steiner reuniu diversas soluções em um trabalho. Por causa deste trabalho, o problema, primeiramente proposto por Fermat, passou a ser conhecido como o *Problema de Steiner*.

Courant e Robbins [11] apresentaram uma maneira de determinar a solução deste problema através de uma construção utilizando régua e compasso. Sejam A, B e C os três pontos, sendo que A é tal que o ângulo formado pelos segmentos $\overline{AB}$ e $\overline{AC}$ é máximo. Temos dois casos para analisar. Se esse ângulo é maior ou igual a 120^o, então a solução é dada pelos segmentos $\overline{AB}$ e $\overline{AC}$. Caso contrário, seja P o ponto do plano euclidiano tal que os segmentos $\overline{AP}$, $\overline{BP}$ e $\overline{CP}$ formam, dois a dois, um ângulo de 120^o. Esses segmentos formam a solução do problema. Vejam as figuras abaixo.

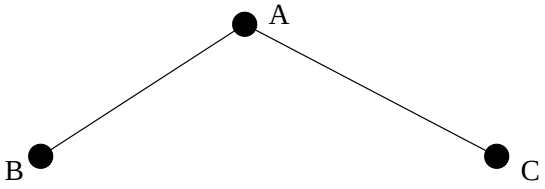

Figura 1: $\hat{A} \geq 120^o$

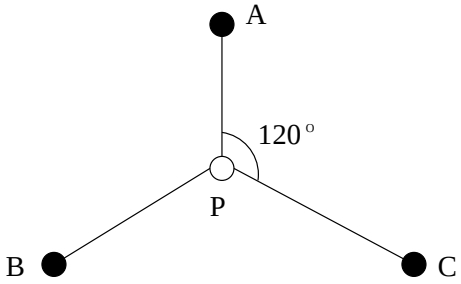

Figura 2: $\hat{A} < 120^o$

O Problema de Steiner em Grafos (PSG) é uma variante do Problema de Steiner. Neste trabalho, apresentaremos uma resenha dos algoritmos de aproximação para o PSG baseados em *árvores k-restritas* (definidas no capítulo 1). O uso de árvores k-restritas mostrou-se eficaz para o PSG — os algoritmos de aproximação com as melhores razões de aproximação conhecidas para o PSG utilizam essas árvores.

Este trabalho está organizado da seguinte forma. No primeiro capítulo, definimos o Problema de Steiner em Grafos, discutimos a sua complexidade computacional e comentamos algumas técnicas de solução do problema. Além disso, estabelecemos a notação que será usada na apresentação dos algoritmos e apresentamos o primeiro algoritmo de aproximação para o problema — o ingênuo *algoritmo da árvore geradora mínima* com razão de aproximação 2.

No segundo capítulo, apresentamos o *algoritmo das árvores 3-restritas*, proposto por Zelikovsky [48] em 1993. Durante vários anos, 2 foi a melhor razão de aproximação constante conhecida para o problema. O algoritmo das árvores 3-restritas foi o primeiro a conseguir uma razão de aproximação constante menor que 2. O algoritmo é simples porém sua análise é complexa e o artigo original continha falhas que foram explicitamente consertadas por Gröpl, Hougardy, Nierhoff e Prömel [21].

No terceiro capítulo, apresentamos o *algoritmo de Berman e Ramayer* [6], que utiliza árvores k-restritas para uma constante $k \geq 3$. No quarto capítulo, descrevemos o *algoritmo do ganho relativo*, elaborado por Zelikovsky [49], que introduziu uma nova maneira de escolher árvores k-restritas. No quinto capítulo, temos o *algoritmo do pré-processamento*, proposto por Karpinski e Zelikovsky [31], que introduziu o conceito de perda de uma árvore de Steiner.

No sexto capítulo apresentamos o *algoritmo de Robins e Zelikovsky* [42], proposto em 2000, cuja razão de aproximação é a melhor conhecida para o PSG até o presente. O último capítulo contém algumas considerações finais.

Esses algoritmos evoluiram uns dos outros e mostram diferentes maneiras de se usar árvores k-restritas no projeto de algoritmos de aproximação para o PSG. É importante ressaltar que o nosso objetivo principal no estudo desses algoritmos é a demonstração de uma razão de aproximação e não a determinação precisa do tempo de execução desses algoritmos. Esperamos que esta dissertação facilite a compreensão desses interessantes algoritmos e sirva como ponto de partida e de apoio para o estudo do PSG e de problemas correlatos.

Capítulo 1

Definições específicas e notação

Neste capítulo descrevemos alguns dos conceitos utilizados neste trabalho, definimos formalmente o *Problema de Steiner em Grafos* e discutimos algumas de suas principais características.

1.1 Teoria dos grafos

Os conceitos e definições abaixo foram retirados dos livros de Bondy e Murty [8] e de Diestel [12].

Um *grafo* $G = (V, E)$ consiste de um conjunto finito e não-vazio V_G e de um conjunto E_G de pares não-ordenados de elementos de V_G. Os elementos de V_G são chamados de *vértices de* G e os de E_G são as *arestas de* G.

Se $e = \{u, v\} \in E_G$, então os vértices u e v são os *extremos* de e. Algumas vezes denotamos e simplesmente por uv. Dois vértices u, v em V_G são *adjacentes* se existir uma aresta com extremos u e v. Duas arestas são *paralelas* se têm os mesmos extremos. Um *laço* é uma aresta que tem os extremos iguais.

Um grafo G é *simples* se ele não tem arestas paralelas e nem laços. Se cada vértice de G é adjacente a todos os outros, então G é *completo*.

Se $A \subseteq V_G$, denotamos por $\delta(A)$ o conjunto de todas as arestas de G que têm um extremo em A e o outro em $V_G \setminus A$. Se $u \in A$ e $v \notin A$, dizemos que $\delta(A)$ *separa* u *e* v. Chamamos tal conjunto de *corte* de G definido por A. Se $A = \{v\}$, escrevemos $\delta(v)$ em vez de $\delta(\{v\})$. O número $|\delta(v)|$ é o *grau de* v.

Se os grafos H e G são tais que $V_H \subseteq V_G$ e $E_H \subseteq E_G$, então dizemos que H é um *subgrafo de* G. Se H é um subgrafo de G e E_H é o conjunto de todas as arestas de G que têm ambos os extremos em V_H, então H é um *subgrafo induzido de* G. Se $V_H = V_G$, dizemos que H é um *subgrafo gerador de* G. Se A é um conjunto de vértices de G, o subgrafo induzido de G que tem A por conjunto de vértices é denotado por $G[A]$.

Seja F um conjunto de pares não-ordenados de vértices de G. Denotamos por $G + F$ o grafo $(V_G, E_G \cup F)$. De modo análogo, $G - F$ é o grafo $(V_G, E_G \setminus F)$. Se $F = \{f\}$,

escrevemos $G + f$ e $G - f$, respectivamente, para $G + \{f\}$ e $G - \{f\}$.

Um *passeio* é uma seqüência $C = (v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$ de elementos de um grafo G onde $v_0, \ldots, v_k$ são vértices e cada e_i é uma aresta com extremos v_{i-1} e v_i. Os vértices v_0 e v_k são os *extremos de* C. Para simplificar, podemos omitir as arestas e representar um passeio por sua seqüência de vértices, $(v_0, \ldots, v_k)$. Algumas vezes interpretamos C como o conjunto de arestas de C. O passeio é *fechado* se $v_0 = v_k$ e *aberto* caso contrário. Se as arestas de C são todas distintas, então C é chamado de *trilha*. Se os vértices de C são todos distintos, então C é chamado de *caminho*. Se $k \geq 1$ e os vértices $v_1, \ldots, v_k$ são todos distintos e $v_0 = v_k$, então C é um *circuito*.

Um *grafo euleriano* G é um grafo onde o grau de todos os vértices é par. Uma *trilha euleriana* P em G é um passeio fechado que passa uma única vez em cada aresta de G. Um *circuito hamiltoniano* C em um grafo G é um passeio fechado que passa exatamente uma vez em cada vértice de G.

Um grafo G é *conexo* se, para cada par u, v de vértices, existe um caminho em G com extremos u e v. Um *componente* de G é um subgrafo conexo maximal de G. Dizemos que dois vértices u e v de G estão *conectados* ou que u está *conectado* a v em G se u e v estão em um mesmo componente de G. De modo análogo, dizemos que um conjunto de vértices X de G está conectado, ou G *conecta* X, se todos os vértices de X estão em um mesmo componente de G. Uma *floresta* é um grafo simples sem circuitos. Uma *árvore* é uma floresta conexa. Algumas vezes confundiremos uma árvore com o seu conjunto de arestas e um componente de um grafo com seu conjunto de vértices. Uma *árvore enraizada em* v é uma árvore onde todas as arestas da árvore têm uma orientação no sentido do vértice v. Esta orientação é obtida através de todos os caminhos em M de um vértice u ao vértice v, onde $u \neq v$. O vértice v é denominado *raiz da árvore*. A figura 1.1 ilustra algumas das definições acima mencionadas.

Dados um grafo G e uma função c que associa um número a cada aresta de G, denotamos por $c(F)$ a soma $\sum_{e \in F} c(e)$ para todo conjunto F de arestas de G. Quando a função c está subentendida, chamamos o número $c(F)$ de *custo de* F e o número $c(e)$ de *custo de* e. Para um subgrafo H de G, escrevemos $c(H)$ em vez de $c(E_H)$ e chamamos este número de *custo de* H.

Se G é completo, dizemos que c *satisfaz a desigualdade triangular* se

$$c(ij) \leq c(ik) + c(kj)$$

para quaisquer três vértices distintos i, j e k de G.

Uma *árvore geradora mínima de* (G, c) é uma árvore geradora em G de custo mínimo. Ou seja, M^* é uma árvore geradora mínima se para toda árvore geradora M em G tem-se que $c(M^*) \leq c(M)$. Denotamos o custo de M^* por $mst(G, c)$.

Seja W um conjunto de vértices de G. A *redução* de c em W é a função que coincide com c exceto nas arestas cujas duas extremidade estão em W, onde ela vale zero.

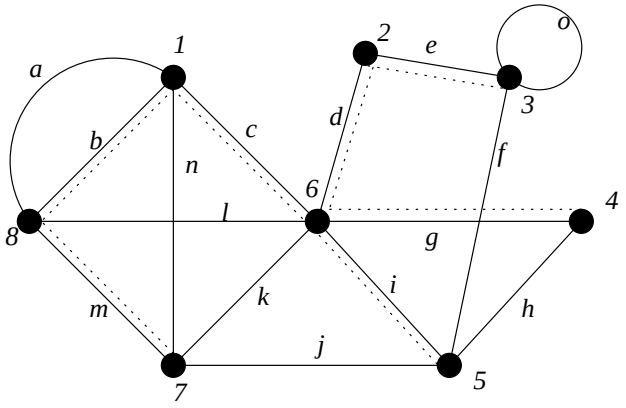

Figura 1.1: Sejam $G = (V, E)$, onde $V := \{1, 2, 3, 4, 5, 6, 7, 8\}$ e $E := \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o\}$. A aresta o é um exemplo de laço e as arestas a e b são paralelas. Se $A := \{4, 5\}$, então $\delta(A) := \{f, g, i, j\}$. A seqüência $P := (1, a, 8, m, 7, k, 6)$ é um exemplo de um caminho em G de 1 a 6. O grafo $G - \{a, o\}$ é simples e as arestas pontilhadas representam as arestas de uma árvore geradora de G.

Denotamos a redução de c em W por $c[W]$. Mais especificamente,

$$c[W](e) = \begin{cases} c(e) & \text{se } e \notin E_{G[W]}, \\ \\ 0 & \text{se } e \in E_{G[W]}. \end{cases}$$

A *restrição* de c a $G[W]$ é a função que coincide com c exceto nas arestas que não estão em $E_{G[W]}$, onde ela não está definida. Mais especificamente, se $\dot{c}$ é a restrição de c a $G[W]$, então

$$\dot{c}(e) = c(e) \quad \text{para toda aresta } e \text{ em } E_{G[W]}.$$

Algumas vezes usamos c no lugar da restrição de c a $G[W]$.

Árvores de Steiner e árvores k-restritas

Considere um grafo completo G, uma função c de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e um conjunto R não-vazio de vértices de G. Chamamos os vértices de R de *terminais* e os de $V_G \setminus R$ de *vértices de Steiner*. Uma árvore T em G é uma *árvore de Steiner de* (G, R) se $R \subseteq V_T$. Uma *árvore de Steiner mínima de* (G, c, R) é uma árvore de Steiner de (G, R) de custo mínimo, ou seja, T^* é uma árvore de Steiner mínima de (G, c, R) se $c(T^*) \leq c(T)$ para toda árvore de Steiner T de (G, R). Denotamos por $smt(G, c, R)$ o custo de uma árvore de Steiner mínima de (G, c, R).

Uma *k-árvore de (G, R)* é uma árvore em G com k folhas cujos vértices internos são de Steiner (ou seja, não estão em R) e as folhas são terminais (estão em R). Para uma k-árvore T_k de (G, R), denotamos por $VT(T_k)$ o conjunto dos terminais de T_k e por $VS(T_k)$ o conjunto dos vértices de Steiner de T_k.

Seja T uma árvore de Steiner de (G, R). Se T' é uma subárvore maximal de T que é uma i-árvore de (G, R) para algum i, dizemos que T' é um *componente cheio de T*. Se, para algum k, todo componente cheio de T é uma i-árvore em (G, R) com $i \leq k$, dizemos que T é uma *árvore k-restrita de (G, R)*.

Uma *árvore k-restrita mínima de (G, c, R)* é uma árvore k-restrita de (G, R) de custo mínimo, ou seja, T^* é uma árvore k-restrita mínima de (G, c, R) se $c(T^*) \leq c(T)$ para toda árvore k-restrita T de (G, R). Denotamos por $smt_k(G, c, R)$ o custo de uma árvore k-restrita mínima de (G, c, R). A figura 1.2 ilustra as definições acima.

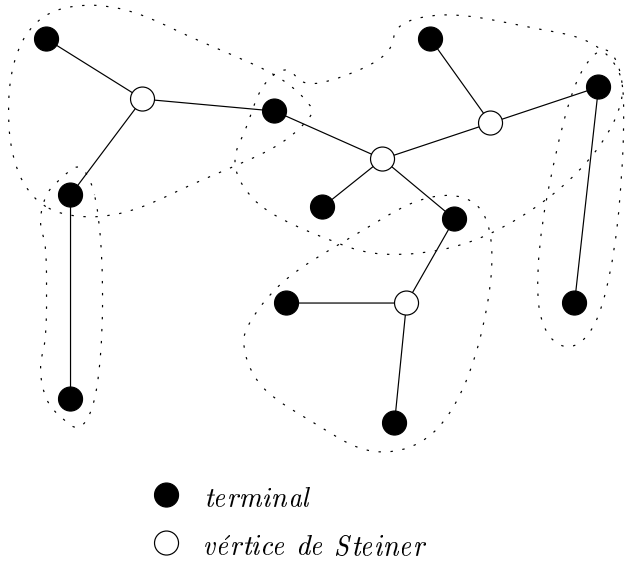

Figura 1.2: Exemplo de árvore de Steiner. Cada componente cheio está destacado pelas linhas pontilhadas. Note que, nesse exemplo, o número máximo de terminais de um componente cheio é 5. Portanto trata-se de uma árvore 5-restrita.

Note que, se uma árvore de Steiner mínima T de (G, c, R) é uma árvore k-restrita, então T é uma árvore k-restrita mínima. Entretanto, o inverso não é verdadeiro. Por exemplo, uma árvore 2-restrita mínima, que é uma árvore geradora mínima no subgrafo induzido pelos terminais, nem sempre é uma árvore de Steiner mínima. Veja a figura 1.3.

Conjunto remoção e função projeção

Sejam H um grafo e $\dot{c}$ uma função de E_H em $Q_{>0}$. Considere uma árvore geradora mínima M de $(H, \dot{c})$ e um conjunto τ de vértices de H. Um *separador de τ em M* é

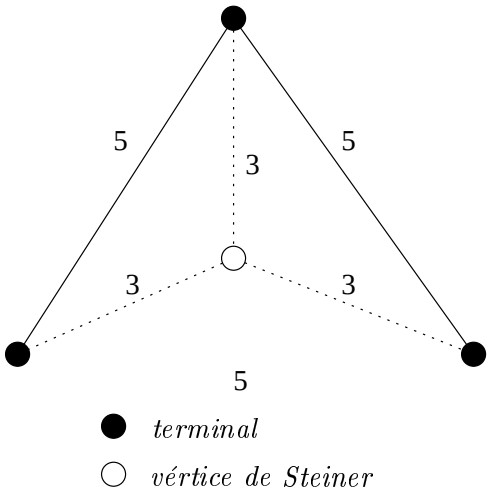

Figura 1.3: No grafo as linhas normais são arestas de uma árvore geradora mínima sobre os terminais e as linhas pontilhadas são as aresta de uma árvore de Steiner mínima.

um conjunto minimal de aresta de M cuja remoção desconecta todo par de vértices de τ. Um *conjunto remoção D de τ em M* é um separador de τ em M com $\dot{c}(D)$ máximo. Denotamos por $ind(M, \dot{c}, \tau)$ o número $\dot{c}(D)$, onde D é um conjunto remoção de τ em M.

Se $\tau = \{x, y\}$, então $ind(M, \dot{c}, \tau)$ é o custo de uma aresta de custo máximo no caminho em M entre os extremos de xy e é denotado por $ind(M, \dot{c}, xy)$. Neste caso, chamamos uma tal aresta de *índice* de xy em M. Note que $\dot{c}(xy) \geq ind(M, \dot{c}, xy)$ para qualquer aresta xy, porque, se $\dot{c}(xy) < ind(M, \dot{c}, xy)$, então uma aresta de custo máximo no caminho em M entre x e y poderia ser substituída pela aresta xy resultando em uma árvore geradora M' de $(H, \dot{c})$ tal que $\dot{c}(M') < \dot{c}(M)$. Um absurdo já que M é uma árvore geradora mínima de $(H, \dot{c})$.

O custo de um conjunto remoção de τ em M também pode ser definido a partir da redução de $\dot{c}$ em τ. Veja a seguinte expressão,

$$ind(M, \dot{c}, \tau) = mst(H, \dot{c}) - mst(H, \dot{c}[\tau]). \tag{1.1}$$

Isso, em particular, mostra que a função índice na verdade não depende da árvore geradora mínima M. Porém, como em geral associamos o valor do índice ao custo de um conjunto remoção, é conveniente manter M como parâmetro.

Seja D um conjunto remoção τ em M. A *função projeção p de D em τ relativa a M* é uma função que associa cada aresta de D uma aresta de H, definida do seguinte modo. Para cada aresta e de D, existem dois componentes de $M - D$, digamos X_e e Y_e, cada um contendo um dos extremos de e. Sejam x e y os vértices de τ em X_e e Y_e respectivamente (cada componente de $M - D$ tem exatamente um vértice de τ, pois D é minimal). A função projeção associa à aresta e a aresta xy. Observe que a imagem da função projeção

é uma árvore geradora de $H[\tau]$. Veja na figura 1.4 um conjunto remoção D de τ em M e a imagem da função projeção p de D em τ relativa a M.

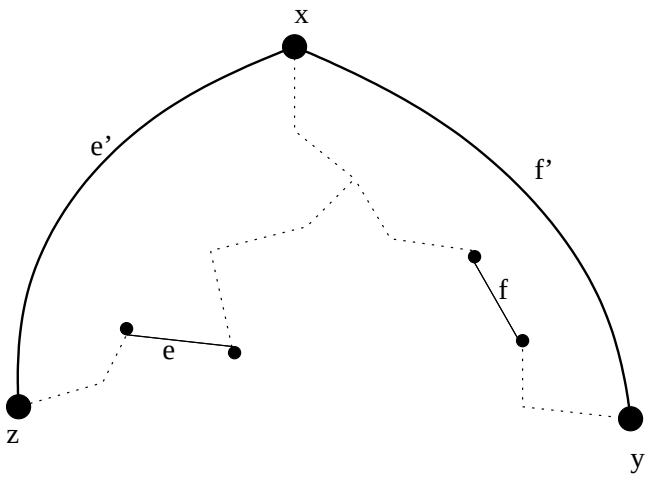

Figura 1.4: O conjunto remoção $D := \{e, f\}$ e a imagem da função projeção $p(D) := \{e', f'\}$ sobre $\tau := \{x, y, z\}$ em M, onde M são as arestas pontilhadas mais as arestas de D.

Vejamos algumas propriedades sobre o custo de um conjunto remoção.

Lema 1.1 *Sejam H um grafo, $\dot{c}$ uma função de E_H em $Q_{>0}$ e M uma árvore geradora mínima de $(H, \dot{c})$. Para todo conjunto τ de vértices de H com $|\tau| \geq 3$, temos que*

$$ind(M, \dot{c}, \tau) \leq ind(M, \dot{c}, \tau \setminus \{v\}) + ind(M, \dot{c}, uv),$$

para qualquer v e u em τ, onde $u \neq v$. Além disso, a igualdade vale para algum u em τ, $u \neq v$.

Prova. Seja D o conjunto remoção de τ em M. Em $M - D$ existem $|\tau|$ componentes, cada um deles contendo um vértice de τ. Seja A o componente que contém v. Seja e uma aresta de custo mínimo em D que conecta A a um outro componente, digamos B. Note que $D \setminus \{e\}$ é um conjunto remoção de $\tau \setminus \{v\}$ em M. Assim, temos a seguinte igualdade

$$ind(M, \dot{c}, \tau) = ind(M, \dot{c}, \tau \setminus \{v\}) + \dot{c}(e).$$

Veja a figura 1.5. Note que, pela escolha de e, temos que $\dot{c}(e) \leq ind(M, \dot{c}, uv)$ para todo u em $\tau \setminus \{v\}$ (e a igualdade vale quando u é o vértice de τ em B). De fato, $\dot{c}(e) \leq \dot{c}(f)$ onde f é a aresta em $D \cap \delta(A)$ que separa u de v, e portanto $\dot{c}(e) \leq ind(M, \dot{c}, uv)$. Portanto

$$ind(M, \dot{c}, \tau) \leq ind(M, \dot{c}, \tau \setminus \{v\}) + ind(M, \dot{c}, uv),$$

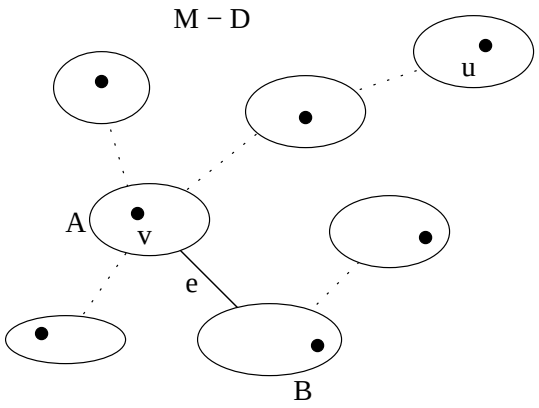

Figura 1.5: As elipses são os componentes de $M - D$ e as linhas pontilhadas são as arestas do conjunto D.

valendo a igualdade para pelo menos um u em $\tau \setminus \{v\}$. □

O lema e o corolário abaixo serão de grande importância na determinação de uma razão de aproximação dos algoritmos que veremos.

Lema 1.2 *Sejam τ_0 e τ_1 conjuntos de vértices de H com pelo menos dois terminais, $\dot{c}$ uma função de E_H em $Q_{>0}$, M uma árvore geradora mínima de $(H, \dot{c})$ e D um conjunto remoção de τ_0 em M e sua respectiva função projeção p relativa a M. Seja $\dot{c}'$ uma função tal que, para toda aresta e de H, tem-se que $\dot{c}'(e) = \dot{c}(e)$ se $e \notin p(D)$ e $\dot{c}'(p(e)) \leq ind(M, \dot{c}, e)$ se $e \in D$. Se $M' := M - D + p(D)$ então*

$$ind(M, \dot{c}, \tau_1) \geq ind(M', \dot{c}', \tau_1).$$

Prova. A prova é por indução em $|\tau_1|$. Primeiramente vamos mostrar que, se $|\tau_1| = 2$, então o lema vale. Sejam x e y os elementos em τ_1.

Sejam P o caminho em M de x a y e P' o caminho em M' de x a y. Se $P \cap D = \emptyset$, então $P = P'$ e não há nada a provar. Se $P \cap D \neq \emptyset$, sejam $e_1, \ldots, e_k$ as arestas de $P \cap D$ e $f_j := p(e_j)$ para $j = 1, \ldots, k$. Seja C_j o único circuito em $M + f_j$.

É fácil ver que $f_j \in P'$ para todo j e que P' consiste de trechos de P e dos C_j, para $j = 1, \ldots, k$. Seja f uma aresta de custo máximo em P', ou seja, f é tal que $\dot{c}'(f) = ind(M', \dot{c}', xy)$. Se $f \in P$, então $ind(M', \dot{c}', xy) = \dot{c}'(f) = \dot{c}(f) \leq ind(M, \dot{c}, xy)$. Se $f \notin P$, então $f \in C_j$ para algum j e $ind(M', \dot{c}', xy) = \dot{c}'(f) \leq \dot{c}(e_j) \leq ind(M, \dot{c}, xy)$.

Agora, suponha que $|\tau_1| > 2$. Pelo lema 1.1, existem u e v em τ_1 onde $u \neq v$ tal que

$$ind(M, \dot{c}, \tau_1) = ind(M, \dot{c}, \tau_1 \setminus \{v\}) + ind(M, \dot{c}, uv).$$

Pela hipótese de indução quando aplicada no lado direito da igualdade acima, temos que

$$\begin{aligned} ind(M, \dot{c}, \tau_1) &\geq ind(M', \dot{c}', \tau_1 \setminus \{v\}) + ind(M', \dot{c}, uv) \\ &\geq ind(M', \dot{c}', \tau_1), \end{aligned}$$

onde a última desigualdade vale pelo lema 1.1. □

Corolário 1.3 *Sejam τ_0 e τ_1 conjuntos não-vazios de vértices de H, $\dot{c}$ uma função de E_H em $Q_{>0}$ satisfazendo a desigualdade triangular e M uma árvore geradora mínima de $(H, \dot{c})$. Se M' é uma árvore geradora mínima de $(H, \dot{c}[\tau_0])$ então*

$$ind(M, \dot{c}, \tau_1) \geq ind(M', \dot{c}[\tau_0], \tau_1).$$

Ganho

Considere novamente um grafo completo G, uma função c de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e um conjunto R não-vazio de vértices de G. Seja T^*_τ uma árvore de Steiner mínima de (G, c, τ), onde τ é um subconjunto de R, $\dot{c}$ é a restrição de c a $G[R]$ e M uma árvore geradora mínima de $(G[R], \dot{c})$. O *ganho* de τ em M indica se é ou não vantajoso utilizar T^*_τ para compor com M uma árvore de Steiner de custo baixo. Matematicamente, o ganho de τ em M é o número $g(M, \tau) := ind(M, \dot{c}, \tau) - smt(G, c, \tau)$. (Fica subentendida a dependência do ganho de G das funções c e $\dot{c}$.) Na figura 1.6 mostramos o ganho de τ em M.

Lema 1.4 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular, R um conjunto não-vazio de vértices de G e τ um subconjunto de R. Considere também $\dot{c}$ e $\dot{c}'$ funções de $E_G[R]$ em $Q_{>0}$, M e M' árvores geradoras mínimas de $(G[R], \dot{c})$ e de $(G[R], \dot{c}')$ respectivamente. Se $ind(M, \dot{c}, \tau) \geq ind(M', \dot{c}', \tau)$, então*

$$g(M, \tau) \geq g(M', \tau).$$

Prova. Segue diretamente da definição de ganho. □

1.2 Problema de Steiner em Grafos

Nesta seção, apresentamos o Problema de Steiner em Grafos, partindo de sua definição até o primeiro algoritmo de aproximação conhecido para o problema. Grande parte das informações desta seção foi retirada da dissertação de mestrado de Ferreira [15].

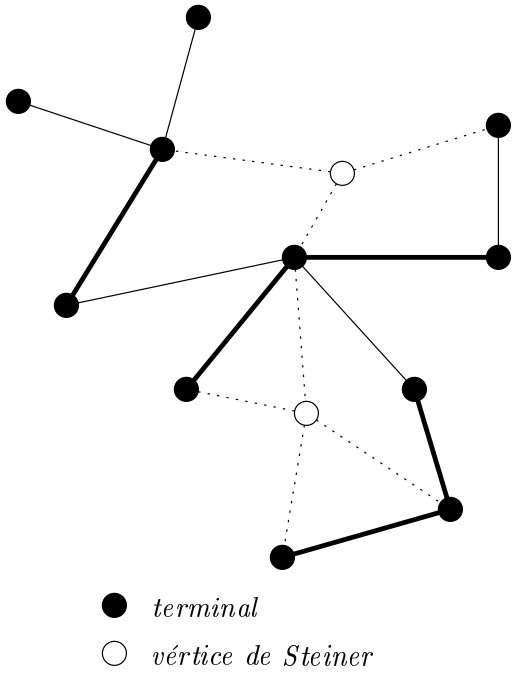

Figura 1.6: As linhas pontilhadas são as arestas de uma árvore de Steiner mínima T^*, as linhas finas juntamente com as grossas representam as arestas de uma árvore geradora M e as linhas grossas representam um conjunto remoção D de $VT(T^*)$ em M. O ganho de $VT(T^*)$ em M é o custo de D menos o custo de T^*.

Definição

Considere um grafo $G = (V, E)$ conexo, uma função c de E_G em $Q_{>0}$ e um conjunto R não-vazio de vértices de G — os terminais. O objetivo do Problema de Steiner em Grafos é encontrar uma árvore de Steiner mínima de (G, c, R).

A primeira versão do Problema de Steiner apareceu como um problema de geometria [35]. A formulação utilizando os elementos da Teoria dos Grafos apareceu, pela primeira vez, em 1971. Ela foi proposta por Hakimi [24] e por Dreyfus e Wagner [14].

Algumas observações podem ser feitas sobre as instâncias do PSG. Se existir em G arestas paralelas, podemos eliminá-las deixando apenas uma aresta de menor custo dentre as arestas paralelas. Laços também podem ser eliminados do grafo. Vértices de Steiner de grau 1 junto com a aresta incidente a este vértice podem ser eliminados.

O *fecho métrico* de (G, c) é o par $(\overline{G}, \overline{c})$, onde $\overline{G}$ é um grafo completo com conjunto de vértices V_G e, para qualquer par de vértices u e v de V_G, $\overline{c}(uv)$ é o custo de um caminho de u até v de custo mínimo de (G, c), ou seja

$$\overline{c}(uv) := \min\{c(P) : P \text{ é um caminho de } u \text{ até } v \text{ em } G\}.$$

Note que, a função $\overline{c}$ satisfaz a desigualdade triangular. Além disso, se $\overline{T}$ é uma árvore de Steiner em $(\overline{G}, \overline{c}, R)$ e T é uma árvore de Steiner de (G, c, R) obtida através da substituição de cada aresta uv de $\overline{T}$ pelas arestas de um caminho mínimo de u a v em G e da remoção de uma aresta de cada eventual circuito criado a partir desta substituição, então $c(T) \leq c(\overline{T})$.

Com base nessas observações, podemos nos restringir a instâncias do PSG que consistam de um grafo completo e uma função custo nas arestas satisfazendo a desigualdade triangular.

Complexidade computacional

Nesta subseção comentaremos sobre a complexidade computacional do PSG e de algumas de suas variantes. Os conceitos sobre complexidade computacional utilizados nessa dissertação podem ser encontrados em [17, 38].

Segundo Karp [30], através do *problema da cobertura exata* consegue-se mostrar que o problema de decisão associado ao PSG é $\mathcal{NP}$*-completo* portanto o PSG é $\mathcal{NP}$-difícil.

Se $|R| = 2$, o PSG é equivalente a encontrar um caminho de custo mínimo entre dois vértices num grafo com custos não-negativos nas arestas. Neste caso, existem algoritmos eficientes (veja, por exemplo [13]). Se $R = V_G$, então o PSG é equivalente a encontrar uma árvore geradora mínima de (G, c). Os dois algoritmos mais conhecidos para encontrar uma árvore geradora mínima em um grafo são os algoritmos de Kruskal [33] e o de Prim [40]. Logo, podemos assumir no restante deste trabalho que $2 < |R| < |V_G|$.

Técnicas de solução

Apesar do PSG ser $\mathcal{NP}$-difícil, muitos algoritmos foram projetados para tentar encontrar uma solução ótima. Nesta subseção, comentaremos de forma bem sucinta alguns desses algoritmos e as técnicas envolvidas.

Um dos primeiros algoritmos para resolver o PSG foi elaborado por Hakimi [24]. Ele utilizou o que denominamos de *enumeração explícita*. Esta técnica se baseia em construir todas as árvores de Steiner e escolher uma de menor custo. A complexidade deste algoritmo é polinomial no número de terminais e exponencial no número de vértices de Steiner. Lawler [34], valendo-se do fato das propriedades do fecho métrico, apresentou um algoritmo cuja complexidade é polinomial no número de vértices de Steiner e exponencial no número de terminais. É importante notar que, se o número de terminais ou o número de vértices de Steiner está fixo, então os algoritmos acima resolvem o PSG em tempo polinomial.

Dreyfus e Wagner [14] utilizaram *programação dinâmica* para resolver o PSG. Shore, Foulds e Gibbons [43] também projetaram um algoritmo que resolve o PSG. Eles usaram *busca exaustiva (branch and bound)* na árvore de soluções.

Uma outra abordagem para resolver o PSG é formulá-lo como um problema de *programação inteira* [1, 3, 4, 35, 37]. Ferreira [15] mostra o resultado da aplicação de ferramentas

da *teoria dos poliedros* ao PSG.

Uma abordagem que tem avançado muito nos últimos anos para problemas $\mathcal{NP}$-difíceis é o projeto de algoritmos de aproximação. É na aplicação dessa abordagem ao PSG que estamos interessados.

Algoritmos de aproximação

Seja (G, c, R) uma instância do Problema de Steiner em Grafos e A um algoritmo que produz uma árvore de Steiner T de (G, R). Dizemos que A é um *algoritmo de aproximação* para o PSG se A consome tempo polinomial no tamanho da instância e $c(T) \leq \alpha \cdot smt(G, c, R)$ para qualquer instância (G, c, R) do PSG, onde α é um número maior do que 1. Neste caso, dizemos que A é uma *α-aproximação* para o PSG e o número α é uma *razão de aproximação* do algoritmo A para o PSG. Em geral, α pode ser uma função dependente da instância do problema, entretanto, neste traballho, estamos interessados em razão de aproximação $O(1)$, ou seja, em razão de aproximação constante. Algumas vezes, para simplicar o texto, usamos o termo algoritmo como sinônimo para algoritmo de aproximação.

O primeiro algoritmo de aproximação para o PSG é atribuído a Moore (1968), de acordo com Gilbert e Pollack [19]. O algoritmo constrói uma árvore geradora mínima de $(G[R], \dot{c})$, onde $\dot{c}$ é a restrição de c a $G[R]$. Este algoritmo, que denotamos por MST, tem uma razão de aproximação de 2 (mais exatamente, de $2 + \frac{1}{n}$, onde n é o número de terminais do grafo). Entre este algoritmo e o algoritmo das árvores 3-restritas outros algoritmos foram projetados [32, 41, 36], porém todos com razão de aproximação de 2. Abaixo, mostramos uma prova da razão de aproximação do algoritmo MST baseada na demostração apresentada por Vazirani [45].

Lema 1.5 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. Se $\dot{c}$ é a restrição de c a $G[R]$, então*

$$mst(G[R], \dot{c}) \leq 2\, smt(G, c, R).$$

Prova. Seja T uma árvore de Steiner mínima de (G, c, R). Vamos construir, a partir de T, uma árvore geradora de $G[R]$ de custo no máximo $2\, c(T)$.

Para isso, duplique as arestas de T, obtendo um grafo euleriano L. Seja P uma trilha euleriana em L; o custo desta trilha é $2\, smt(G, c, R)$. Sem perda de generalidade, podemos assumir que $P := (v_0, e_1, \ldots, e_l, v_l)$, onde $v_0 = v_l$ é terminal e folha em T.

Utilizando P obtemos um circuito hamiltoniano $C := (c_0, e'_1, c_1, \ldots, c_{k-1}, e'_k, c_k)$ em $G[R]$. O circuito C foi obtido de maneira que $c_0 = c_k = v_0$ e para cada par de vértices c_{i-1} e c_i, $1 \leq i \leq k$, existem dois terminais v_x e v_y em P tais que $0 < x < y \leq l$. Além disso, se existe um terminal v_t em P entre v_x e v_y, então $v_t = c_j$ para algum $0 \leq j < i - 1$. Note que C pode ser obtido em tempo polinomial.

Por causa da desigualdade triangular, o custo do circuito hamiltoniano C não é maior do que o custo da trilha euleriana P. Jogando fora uma aresta deste circuito, obtemos uma árvore geradora M de $G[R]$ que tem custo no máximo $2\,smt(G, c, R)$. Portanto, $mst(G[R], \dot{c}) \leq 2\,smt(G, c, R)$. □

A análise da razão de aproximação do algoritmo MST é justa, ou seja, existe uma instância do PSG para o qual o custo da solução produzida pelo algoritmo MST dividido pelo custo da árvore de Steiner mínima é tão próximo de 2 quanto se queira.

Veja na figura 1.7 tal instância. O grafo é composto por n terminais e um vértice de Steiner. As arestas que ligam o vértice de Steiner aos terminais têm custo 1, enquanto que as outras arestas têm custo 2. O custo de uma árvore geradora mínima para essa instância do problema é $2(n-1)$ e o custo de uma árvore de Steiner ótima é n.

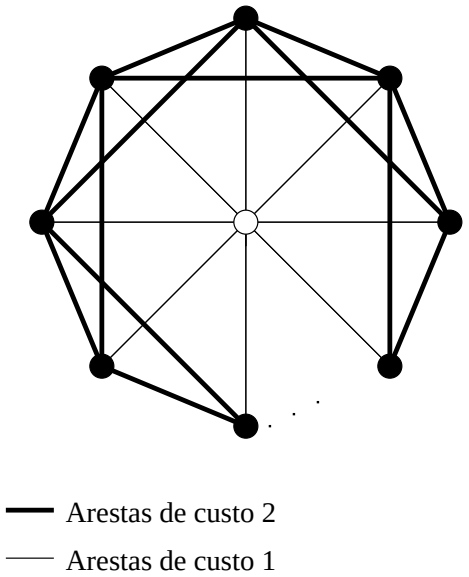

Figura 1.7: Grafo onde a razão de aproximação do algoritmo MST é $2 + \frac{1}{n}$.

Lembrando que uma árvore geradora mínima de $(G[R], \dot{c})$ é uma árvore 2-restrita mínima de (G, c, R), uma maneira de generalizar esse algoritmo, construindo boas aproximações para o PSG seria um algoritmo que devolvesse uma árvore k-restrita mínima para algum $k > 2$ fixo (a figura 1.8 e o lema 1.6 abaixo comprovam a nossa afirmação). Borchers e Du [9] demonstraram uma fórmula para calcular o máximo da razão entre o custo de uma árvore k-restrita mínima e o custo de uma árvore de Steiner mínima para qualquer instância (G, c, R) do PSG.

Lema 1.6 *Seja r_k o supremo da razão entre $smt_k(G, c, R)$ e $smt(G, c, R)$ para quais-*

quer G, c e R. Se $k = 2^l + s$, onde l e s são naturais e $0 \leq s < 2^l$, então

$$r_k = \frac{(l+1)2^l + s}{l2^l + s}.$$

Para uma árvore 2-restrita mínima já vimos um algoritmo eficiente e, para encontrar uma árvore 3-restrita mínima existe uma $(1+\epsilon)$-aproximação probabilística polinomial [39], mas nenhuma demonstração de que o problema é $\mathcal{NP}$-difícil. Em particular, pelo lema 1.6, temos que $r_2 = 2$ e $r_3 = \frac{5}{3}$.

Infelizmente, para $k \geq 4$, construir uma árvore k-restrita mínima é um problema $\mathcal{NP}$-difícil [30]. Contudo, mesmo sendo difícil produzir tais árvores, elas serão utilizadas nos algoritmos e nas provas de razões de aproximações.

Com exceção do algoritmo de Berman e Ramayer os algoritmos que veremos são algoritmos gulosos (detalhes sobre algoritmos gulosos podem ser encontrados em [10]), ou seja, selecionam algumas árvores k-restritas, onde k é uma constante fixa, cujos vértices de Steiner estarão presentes, junto com os terminais, na solução devolvida pelos algoritmos. Além disso, usaremos árvores k-restritas mínimas como limitantes superiores das soluções produzidas pelos algoritmos.

Vale ressaltar que o método guloso, apesar de ser o método utilizado pelo algoritmo com a menor razão de aproximação conhecida até o momento, não é o único para se construir soluções aproximadas para o PSG. Por exemplo, Williamson, Goemans, Mihail e Vazirani [46] projetaram um algoritmo para o PSG baseado no método *primal-dual* [1].

Inaproximabilidade

A menor razão de aproximação conhecida para o PSG é 1,55. Uma pergunta natural é até que ponto podemos continuar procurando algoritmos que consigam melhorar essa razão de aproximação supondo que $\mathcal{P} \neq \mathcal{NP}$. Nesta subseção mostraremos alguns resultados que impõe limites para uma razão de aproximação do PSG.

Arora, Lund, Motwani, Sudan e Szegedy [2] juntamente com uma redução, devida a Bern e Plassman [7], do *problema da B-cobertura de vértices*[2] para o PSG mostram que existe uma constante $c > 1$ tal que nenhum algoritmo de aproximação para o PSG tem razão de aproximação menor do que c, a menos que $\mathcal{P} = \mathcal{NP}$. Denominamos uma tal constante c de *limite de aproximabilidade* (*approximation threshold*).

Além disso, Bern e Plassman mostraram que, se existe um algoritmo com razão de aproximação $1+\epsilon$ para o PSG, então existe um algoritmo para o problema da B-cobertura de vértices com razão de aproximação $1 + (1+B)\,\epsilon$. Portanto, bons algoritmos para o PSG implicam em bons algoritmos para o problema da B-cobertura de vértices. Com esse

[1] Detalhes sobre o método primal-dual podem ser encontrados em [16, 45].

[2] O problema da B-cobertura de vértices consiste em descobrir um conjunto de vértices de custo mínimo tal que toda aresta do grafo tenha um dos extremos nesse conjunto onde o grau máximo dos vértices do grafo é menor ou igual a B.

resultado, se conhecermos um limite de aproximabilidade para o problema da B-cobertura de vértices, então temos um limite de aproximabilidade para o PSG.

Berman e Karpinski [5] mostraram que o melhor limite de aproximabilidade para o problema da 4-cobertura de vértices é de 1,0128. Logo temos um limite de aproximabilidade de 1,0025 para o PSG. Recentemente Thimm [44], utilizando uma redução de Håstad [25] para o problema Max-E3-Lin-2, melhorou este resultado obtendo um valor de 1,0074.

Podemos concluir que, mesmo com os avanços no projeto de algoritmos de aproximação, há uma grande diferença entre a razão de aproximação do melhor algoritmo conhecido até o presente e do melhor limite de aproximabilidade do PSG.

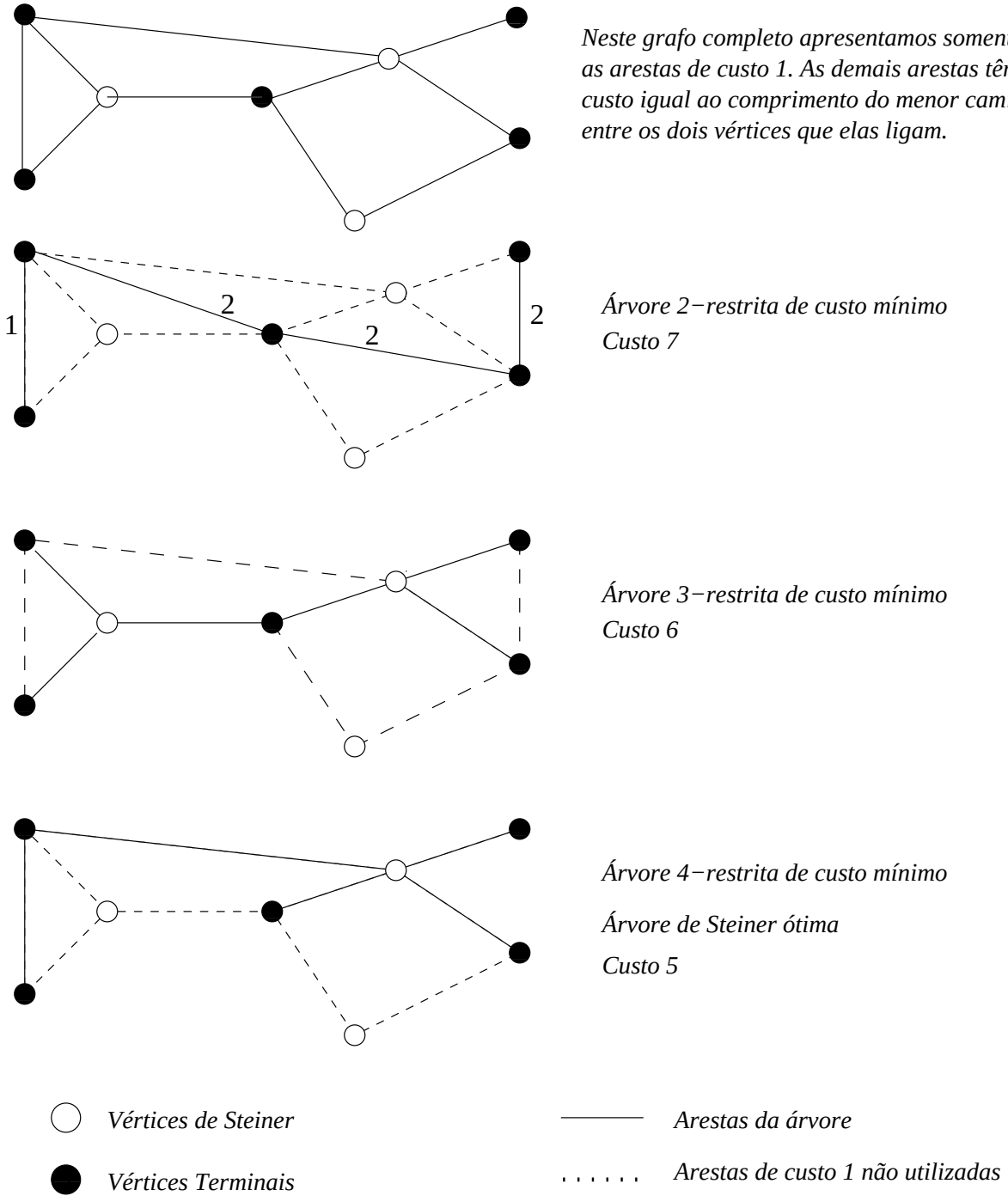

Figura 1.8: Exemplos de árvores k-restritas mínimas.

Capítulo 2

Algoritmo das árvores 3-restritas

O algoritmo das árvores 3-restritas é um algoritmo guloso projetado por Zelikovsky [48] para o problema de Steiner em grafos. Este algoritmo foi o primeiro a superar o ingênuo algoritmo MST, obtendo uma razão de aproximação de $\frac{11}{6} < 1{,}834$. Zelikovsky utilizou árvores 3-restritas mínimas para inserir na solução produzida pelo seu algoritmo alguns vértices de Steiner. Esta abordagem inspirou vários outros algoritmos de aproximação para o PSG. A seguir, descrevemos o algoritmo das árvores 3-restritas.

2.1 Descrição do algoritmo

A entrada do algoritmo é um grafo completo G, uma função c de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e um conjunto R de vértices — os terminais. Como saída, o algoritmo produz uma árvore de Steiner cujo custo é no máximo $\frac{11}{6}$ do custo de uma árvore de Steiner mínima de (G, c, R).

Para um conjunto arbitrário τ de três terminais, que chamaremos de *tripla*, denote por T_τ uma árvore de Steiner mínima de (G, c, τ). Note que, neste caso, T_τ é também uma árvore 3-restrita mínima.

Vamos apresentar primeiramente uma variante do algoritmo de Zelikovsky. Depois comentaremos sobre o algoritmo original. O algoritmo começa com uma árvore geradora mínima M de $(G[R], \dot{c})$, onde $\dot{c}$ é a restrição de c a $G[R]$, e com um subgrafo S de G. Inicialmente S é a árvore M. Em cada iteração, o algoritmo escolhe uma tripla τ que maximiza $g(M, \tau)$. Se $g(M, \tau)$ é positivo, o algoritmo constrói uma árvore de Steiner mínima de (G, c, τ) e um conjunto remoção D_τ de τ em M. Então, o algoritmo diminui o custo $\dot{c}(e)$ de cada aresta e em $G[R]$ que está na imagem da função projeção de D_τ em τ. O custo de cada uma dessas arestas diminui de $g(M, \tau)$. Além disso, o algoritmo altera M e S adequadamente e pára quando não há mais triplas com ganho positivo. Neste ponto, o algoritmo devolve S. Veja a figura 2.1.

Note que o algoritmo das árvores 3-restritas pode ser implementado de forma que seu consumo de tempo seja polinomial no tamanho da sua entrada.

Algoritmo das árvores 3-restritas(G, c, R)
1 Seja $\dot{c}_0$ a restrição de c a $G[R]$;
2 Seja M_0 uma árvore geradora mínima de $(G[R], \dot{c}_0)$;
3 $S_0 := M_0$;
4 $i := 1$;
5 **Enquanto** existir $\tau \subseteq R$, $|\tau| = 3$, com $g(M_{i-1}, \tau) > 0$ **faça**
6 Seja $\tau_i \subseteq R$ com $|\tau_i| = 3$ e $g(M_{i-1}, \tau_i)$ máximo;
7 Seja T_i uma árvore 3-restrita mínima de (G, c, τ_i);
8 Seja D_i um conjunto remoção de τ_i em M_{i-1};
9 Seja p a função projeção de D_i em τ_i relativa a M_{i-1};
10 $A_i := \emptyset$;
11 $c_i := c_{i-1}$;
12 **Para** cada e em D_i **faça**
13 $\dot{c}_i(p(e)) := \dot{c}_{i-1}(e) - g(M_{i-1}, \tau_i)$;
14 $A_i := A_i \cup \{p(e)\}$;
15 $M_i := M_{i-1} - D_i + A_i$;
16 $S_i := S_{i-1} - D_i + T_i$;
17 $i := i + 1$;
18 $f := i - 1$;
19 **Devolva** S_f.

Figura 2.1: Algoritmo das árvores 3-restritas

2.2 Análise do algoritmo das árvores 3-restritas

No artigo original de Zelikovsky [48] havia um erro na análise do algoritmo. Esse erro foi explicitamente corrigido, apenas em 2001, no trabalho de Gröpl, Hougardy, Nierhoff e Prömel [21]. A nossa prova de uma razão de aproximação é baseada nesse trabalho.

A análise a seguir é longa e técnica. Os lemas 2.1 e 2.5 são utilizados na demonstração do teorema 2.6. Eles provam invariantes do algoritmo. O lema 2.2 é usado na demonstração do lema 2.3 e os lemas 2.3 e 2.4 são utilizado na demonstração do lema 2.5.

Lema 2.1 *Para todo $i = 0, \ldots, f$, no algoritmo das árvores 3-restritas temos que $\dot{c}_i(e) \geq 0$ para toda aresta e de $G[R]$, M_i é uma árvore geradora mínima de $(G[R], \dot{c}_i)$ e $\dot{c}_i(M_i) = \dot{c}_0(M_0) - 2\sum_{j=1}^{i} g_{\tau_j}$, onde $g_{\tau_j} := g(M_{j-1}, \tau_j)$.*

Prova. A prova é por indução em i. Para $i = 0$, o lema é trivial. Para $i > 0$, suponha que $\dot{c}_{i-1}$ é não-negativo, que M_{i-1} é uma árvore geradora mínima de $(G[R], \dot{c}_{i-1})$ e que $\dot{c}_{i-1}(M_{i-1}) = \dot{c}_0(M_0) - 2\sum_{j=1}^{i-1} g_{\tau_j}$.

Vamos mostrar que $\dot{c}_i(e) \geq 0$ para toda aresta e de $G[R]$. Note que as alterações de $\dot{c}_{i-1}$ para $\dot{c}_i$ ocorrem na linha 13 do algoritmo. Portanto basta mostrar que o novo custo atribuído é não-negativo.

Digamos que $\tau_i = \{x, y, z\}$, $D_i = \{e, f\}$ e p é a função projeção de τ_i em D_i relativa a M_{i-1}. Sem perda de generalidade, suponha que x é tal que e é a aresta de custo máximo no caminho de x a y em M_{i-1} e f é a aresta de custo máximo no caminho de x a z em M_{i-1}. Se v é o vértice de Steiner de T_i (veja a figura 2.2), então

$$c(xv) + c(vy) \geq c(xy) \geq \dot{c}_{i-1}(xy) \geq \dot{c}_{i-1}(e),$$

onde a primeira desigualdade vale pela desigualdade triangular, a segunda pelo fato que $\dot{c}_{i-1}(e) \geq \dot{c}_i(e)$[1] para toda aresta e de $G[R]$ e a terceira porque M_{i-1} é uma árvore geradora mínima de $(G[R], \dot{c}_{i-1})$. Utilizando a desigualdade acima podemos concluir que

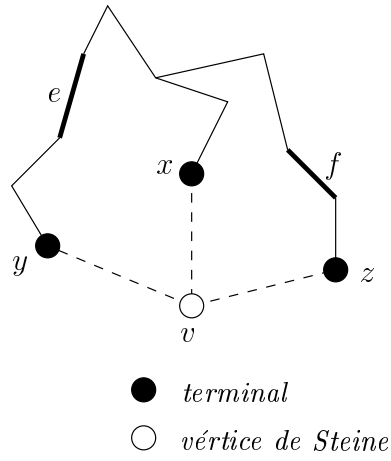

Figura 2.2: Linhas em negrito representam as arestas e e f, linhas normais são arestas de M_{i-1} e linhas pontilhadas são as arestas de uma árvore de 3-retrita mínima de (G, c, τ).

$$\begin{aligned} \dot{c}_{i-1}(f) - g(M_{i-1}, \tau_i) &= \dot{c}_{i-1}(f) - (\dot{c}_{i-1}(e) + \dot{c}_{i-1}(f) - smt(G, c, \tau_i)) \\ &= smt(G, c, \tau_i) - \dot{c}_{i-1}(e) \\ &= c(xv) + c(vy) + c(vz) - \dot{c}_{i-1}(e) > 0. \end{aligned}$$

De forma análoga, podemos concluir que $\dot{c}_{i-1}(f) - g(M_{i-1}, \tau_i) > 0$.

Pela definição de D_i e p, M_i é uma árvore. Além disso, como o custo $\dot{c}_i(p(e)) < \dot{c}_{i-1}(e)$ para cada e em D_i, temos que $p(e)$ é a aresta de custo mínimo de $(G[R], \dot{c}_i)$ no corte definido por uma das componentes de $M_{i-1} - e$. Portanto M_i é uma árvore geradora mínima de $(G[R], \dot{c}_i)$. Quanto ao custo de M_i, temos que $\dot{c}_i(M_i) = \dot{c}_{i-1}(M_{i-1}) - 2\, g_{\tau_i}$. Assim, por indução,

$$\dot{c}_i(M_i) = \dot{c}_0(M_0) - 2 \sum_{j=1}^{i} g_{\tau_j} \quad \square$$

[1]Na linha 13 do algoritmo note que $p(e) = xy$ não está em M_{i-1} e e é uma aresta de custo máximo no caminho de x a y em M_{i-1}, portanto $\dot{c}_{i-1}(p(e)) \geq \dot{c}_{i-1}(e) > \dot{c}_i(p(e))$ pois $g(M_{i-1}, \tau_i) > 0$. Para as outras arestas temos que $\dot{c}_{i-1}(e) \geq \dot{c}_i(e)$.

O próximo lema será usado na demonstração do lema 2.3.

Lema 2.2 *No algoritmo das árvores 3-restritas, se $i \geq 1$, τ é uma tripla, D é um separador de τ em M_i e $\dot{c}_i(D) - smt(G, c, \tau) > 0$, então $D \cap A_i = \emptyset$.*

Prova. Digamos que $A_i := \{e', f'\}$ e $D_i := \{e, f\}$, onde $e' := p(e)$ e $f' := p(f)$. Suponha por contradição que $D \cap A_i \neq \emptyset$. Sem perda de generalidade, podemos assumir que $e' \in D$. Se $f' \in D$ também, então temos que

$$\begin{aligned} \dot{c}_i(D) - smt(G, c, \tau) &= \dot{c}_i(e') + \dot{c}_i(f') - smt(G, c, \tau) \\ &= \dot{c}_{i-1}(e) + \dot{c}_{i-1}(f) - smt(G, c, \tau) - 2\, g(M_{i-1}, \tau_i) \\ &\leq g(M_{i-1}, \tau) - 2\, g(M_{i-1}, \tau_i) \\ &< 0, \end{aligned}$$

pois e e f são separadores de τ em M_{i-1}, $g(M_{i-1}, \tau_i) \geq g(M_{i-1}, \tau)$ e $g(M_{i-1}, \tau_i) > 0$.

Por outro lado, se $f' \notin D$, seja h o elemento de D distinto de e'. Temos que

$$\begin{aligned} \dot{c}_i(D) - smt(G, c, \tau) &= \dot{c}_i(e') + \dot{c}_i(h) - smt(G, c, \tau) \\ &= \dot{c}_{i-1}(e) + \dot{c}_{i-1}(h) - smt(G, c, \tau) - g(M_{i-1}, \tau_i). \end{aligned}$$

Se $\{e, h\}$ é um separador de τ em M_{i-1}, então

$$\dot{c}_i(D) - smt(G, c, \tau) \leq g(M_{i-1}, \tau) - g(M_{i-1}, \tau_i) \leq 0.$$

Já se $\{e, h\}$ não é um separador de τ em M_{i-1}, então seja C o único circuito em $M_i + f$. Considere o par de vértices x e y de τ tal que h é a única aresta de D no caminho P de x a y em M_i. Se $\{e, h\}$ não é um separador então P contém f; do contrário P seria o caminho de x a y em M_{i-1}. O caminho P' de x a y em M_{i-1} contém uma aresta de C se e só se P não a contém. Em particular, se P' não contém h então h está em C. Neste caso, $\{e, f\}$ é um separador de τ em M_{i-1} e $\dot{c}_{i-1}(f) \geq \dot{c}_{i-1}(h)$. Veja a figura 2.3. Assim, temos que

$$\begin{aligned} \dot{c}_i(D) - smt(G, c, \tau) &\leq \dot{c}_{i-1}(e) + \dot{c}_{i-1}(f) - smt(G, c, \tau) - g(M_{i-1}, \tau_i) \\ &\leq g(M_{i-1}, \tau) - g(M_{i-1}, \tau_i) \\ &\leq 0. \end{aligned}$$

Nos vários casos, chegamos a uma contradição da hipótese que $\dot{c}_i(D) - smt(G, c, \tau) > 0$. Portanto podemos concluir que $D \cap A_i = \emptyset$. □

O lema 2.3 abaixo, mais técnico, é uma peça central na demonstração de que, em cada iteração do algoritmo, todas as arestas do conjunto D_i estão em S_{i-1}. Isso será usado na prova de que S_f contém uma árvore de Steiner de (G, R).

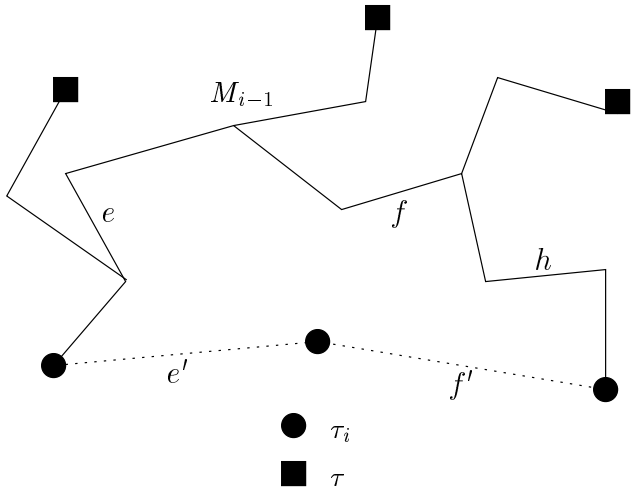

Figura 2.3: Temos que $M_i := M_{i-1} - \{e, f\} + \{e', f'\}$, $\{e', h\}$ é um separador de τ em M_i, $\{e, h\}$ não é um separador de τ em M_{i-1} mas $\{e, f\}$ é.

Lema 2.3 *Sejam $i \geq 1$ e a em M_i. Se τ é uma tripla tal que $D := \{a, b\}$ é um separador de τ em M_i para algum b e $\dot{c}_i(D) - smt(G, c, \tau) > 0$, então $a \in M_{i-1}$ e existe uma tripla τ' e uma aresta b' em M_{i-1} tal que $D' := \{a, b'\}$ é um separador de τ' em M_{i-1} e $\dot{c}_{i-1}(D') - smt(G, c, \tau') \geq \dot{c}_i(D) - smt(G, c, \tau)$.*

Prova. Sejam $D_i := \{e, f\}$ um separador de τ_i em M_{i-1} e $A_i := \{e', f'\}$ a imagem da função projeção de D_i de τ_i relativa a M_{i-1}. Pelo lema 2.2, $D \cap A_i = \emptyset$, ou seja, $\{a, b\} \subseteq M_{i-1}$. Sejam A, B e C os componentes de $M_i - \{e', f'\}$ como na figura 2.4. Sejam C_1 o único circuito em $M_i + e$ e C_2 o único circuito em $M_i + f$.

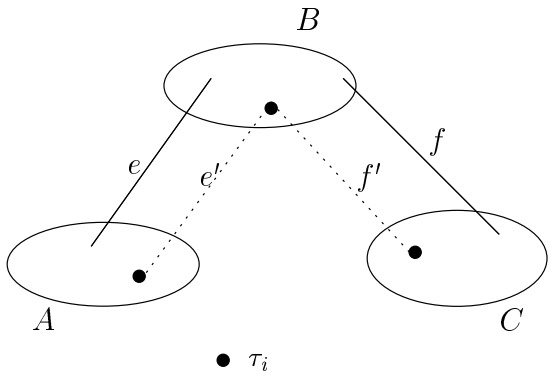

Figura 2.4:

Caso 1: $a \notin C_1 \cup C_2$

Sem perda de generalidade, suponha que $a \in E_A$ (de forma análoga vale para E_B e E_C). Sejam A' e A'' os componentes de $A - a$ e suponha que A'' não contém nenhum extremo de e' e f'. Note que A'' também não contém os extremos de e ou f e contém pelo menos um e no máximo dois elementos de τ pois $\{a, b\}$ é um separador de τ em M_i. Veja a figura 2.5. Se A'' contém dois elementos, então $\{a, b\}$

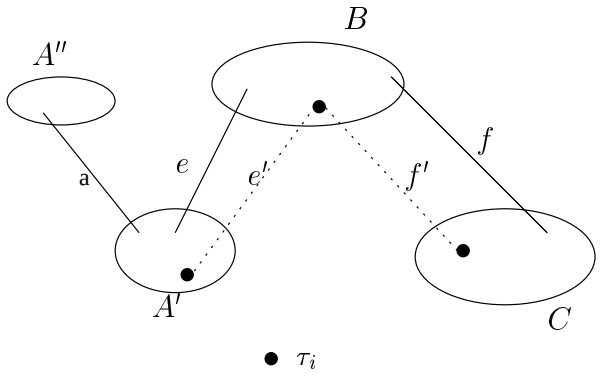

Figura 2.5:

é um separador de τ em M_{i-1}. Assim, tome $\tau' := \tau$ e $b' := b$ e vale o lema.

Se A'' contém exatamente um elemento de τ e os outros dois elementos de τ estão em A' ou B ou C, então $\{a, b\}$ é um separador de τ em M_{i-1} e o lema é verdadeiro. Senão, apenas um entre A', B e C não contém elementos de τ. Neste caso, se $\{a, b\}$ não é um separador de τ em M_{i-1}, então ou e ou f ou ambos estão no caminho entre esses vértices. Tome $\tau' := \tau$ e b' de custo máximo entre os elementos de $\{e, f\}$ que estão no caminho entre esses vértices. Note que $\dot{c}(b') \geq \dot{c}(b)$, pois b está em algum dos circuitos C_1 e C_2. Assim, temos que

$$\begin{aligned} \dot{c}_{i-1}(D') - smt(G, c, \tau') &= \dot{c}_{i-1}(a) + \dot{c}_{i-1}(b') - smt(G, c, \tau) \\ &\geq \dot{c}_{i-1}(a) + \dot{c}_{i-1}(b) - smt(G, c, \tau) \\ &= \dot{c}_i + \dot{c}_i(b) - smt(G, c, \tau) \\ &= \dot{c}_i(D) - smt(G, c, \tau). \end{aligned}$$

Caso 2: $a \in C_1 \cup C_2$

Caso 2.1: $a \in E_A$ ($a \in E_C$ é análogo)

Sejam A' e A'' os componentes de $A - a$ onde A' contém um extremo de e'. Note que A'' contém pelo menos um e no máximo dois elementos de τ pois $\{a, b\}$ é um separador de τ em M_i. Veja a figura 2.6. Se A'' tem dois elementos

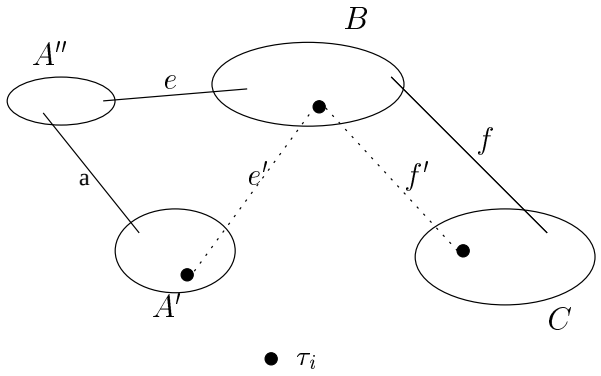

Figura 2.6:

de τ e A' tem um ou A' tem dois elementos de τ, então $\{a, b\}$ é um separador de τ em M_{i-1}. Caso contrário há pelo menos um elemento de τ em $B \cup C$. Se A' contém um elemento de τ, então, supondo que b não separa τ em M_{i-1}, ou e ou f ou ambos separam em M_{i-1} um par de elementos de τ. Tome $\tau' := \tau$ e b' em $\{e, f\}$ que separa τ e tem custo máximo. De modo análogo ao caso 1, temos que $\dot{c}(b') \geq \dot{c}(b)$ e o lema vale. Se A' não contém nenhum elemento de τ, então a não separa nenhum elemento de τ em M_{i-1} e $\{e, b\}$ é um separador de τ em M_{i-1}. Assim, tome $\tau' := \tau_i$ e $b' := f$,

$$\begin{aligned}
\dot{c}_{i-1}(D') - smt(G, c, \tau') &= \dot{c}_{i-1}(a) + \dot{c}_{i-1}(f) - smt(G, c, \tau_i) + \dot{c}_{i-1}(e) - \dot{c}_{i-1}(e) \\
&= \dot{c}_{i-1}(D_i) - smt(G, c, \tau_i) + \dot{c}_i(a) - \dot{c}_{i-1}(e) \\
&= g(M_{i-1}, \tau_i) + \dot{c}_i(a) - \dot{c}_{i-1}(e) \\
&\geq g(M_{i-1}, \tau_i) \qquad (2.1) \\
&\geq g(M_{i-1}, \tau) \\
&= \dot{c}_{i-1}(a) + \dot{c}_{i-1}(b) - smt(G, c, \tau) \\
&= \dot{c}_i(a) + \dot{c}_i(b) - smt(G, c, \tau) \\
&= \dot{c}_i(D) - smt(G, c, \tau),
\end{aligned}$$

onde a desigualdade (2.1) vale pois a está no caminho em M_{i-1} entre o vértice de τ_i que está em A' e o vértice de τ_i que está em B e $\dot{c}_{i-1}(a) \leq \dot{c}_{i-1}(e)$.

Caso 2.2: $a \in E_B$

Sejam B' e B'' os componentes de $B - a$ onde B' contém o extremo comum de e' e f'. Note que B'' contém pelo menos um e no máximo dois elementos de τ. Veja a figura 2.7. Se B'' tem dois elementos de τ e B' tem um elemento de τ, então $\{a, b\}$ é um separador de τ em M_{i-1}. Do mesmo modo, se B'' tem um elemento de τ e B' ou A ou C tem dois elementos de τ, então $\{a, b\}$ é um separador de τ em M_{i-1}.

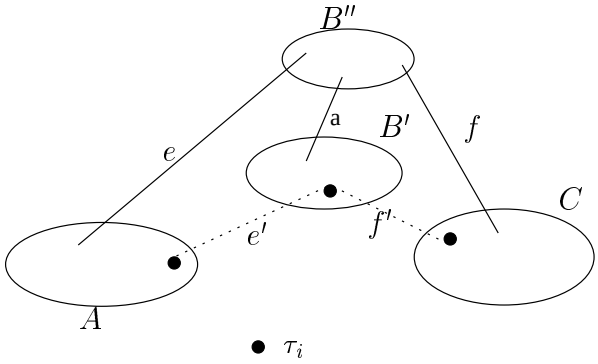

Figura 2.7:

Senão há pelo menos um elemento de τ em $A \cup C$. Se $A \cup C$ contém exatamente um elemento de τ, então tome $\tau' := \tau$ e $b' := e$ ou f dependendo desse elemento estar em A ou C. Note que neste caso b está em C_1 ou C_2, assim $\dot{c}_{i-1}(b') \geq \dot{c}_{i-1}(b)$ e vale o lema (análoga à uma situação do caso 1).

Se $A \cup C$ contém dois elementos de τ, então a não separa nenhum elemento de τ em M_{i-1}, logo tome $\tau' := \tau_i$ e b' em $\{e, f\}$ de custo máximo em M_{i-1}. Como no caso 1, o lema vale pois, como no caso 2.1, temos que $\dot{c}_{i-1}(D') - smt(G, c, \tau') \geq \dot{c}_i(D) - smt(G, c, \tau)$

Pelos casos 1 e 2 podemos concluir que o lema é verdadeiro. □

Finalmente, o lema abaixo garante que as arestas dos conjuntos remoção escolhidos pelo algoritmo não tiveram o seu custo alterado durante a execução do algoritmo, ou seja, não fizeram parte de nenhum conjunto A_i de uma iteração anterior.

Lema 2.4 *Se $i \geq 0$, τ é uma tripla, D é um conjunto separador de τ em M_i e $\dot{c}_i(D) - smt(G, c, \tau) > 0$, então $D \cap \bigcup_{j=1}^{i} A_j = \emptyset$.*

Prova. A prova é por indução em i. Para $i = 0$ o lema é trivial. Por indução, suponha que o lema vale para $i - 1$. Agora provaremos que o lema vale para i.

Primeiramente, pelo lema 2.2, $D \cap A_i = \emptyset$. Agora, seja a um elemento arbitrário de D. Vamos mostrar que $a \notin \bigcup_{j=1}^{i-1} A_j$. Como $i \geq 1$, D é um separador de τ em M_i, $a \in D$ e $\dot{c}_i(D) - smt(G, c, \tau) > 0$, pelo lema 2.3, $a \in M_{i-1}$ e existe b em M_{i-1} e uma tripla τ' tal que $D' := \{a, b'\}$ é um separador de τ' em M_{i-1} com $\dot{c}_{i-1}(D') - smt(G, c, \tau') > 0$. Mas então, pela hipótese de indução, $D' \cap \bigcup_{j=1}^{i-1} A_j = \emptyset$, ou seja, $a \notin \bigcup_{j=1}^{i-1} A_j$. Como isso vale para um elemento arbitrário de D, concluímos que $D \cap \bigcup_{j=1}^{i-1} A_j = \emptyset$ e portanto $D \cap \bigcup_{j=1}^{i} A_j = \emptyset$. □

A seguir mostraremos que S_f contém uma árvore de Steiner de (G, R) e que seu custo não passa de $c(S_0) - \sum_{j=1}^{f} g(M_{j-1}, \tau_j)$.

Para tanto, considere uma variante do algoritmo em que, para cada árvore T_i escolhida pelo algoritmo na linha 7, é feita uma cópia do vértice de Steiner de T_i. Com essas cópias, podemos assumir que T_i utiliza um vértice de Steiner nunca antes utilizado (eventualmente tal vértice será uma cópia de um vértice utilizado anteriormente). Para essa variante do algoritmo, podemos mostrar o seguinte lema.

Lema 2.5 *Para todo i, $0 \leq i \leq f$, no algoritmo das árvores 3-restritas temos que S_i é uma árvore de Steiner de (G, R), $c(S_i) = c(S_0) - \sum_{j=1}^{i} g(M_{j-1}, \tau_j)$ e cada aresta de $M_i \cap S_i$ separa os mesmos pares de terminais em M_i e S_i.*

Prova. A prova é por indução em i. Para $i = 0$ o lema é trivial. Para $i > 0$, suponha que S_{i-1} é uma árvore de Steiner de (G, R), que $c(S_{i-1}) = c(S_0) - \sum_{j=1}^{i-1} g(M_{j-1}, \tau_j)$ e que cada aresta de $M_{i-1} \cap S_{i-1}$ separa os mesmos pares de terminais em M_{i-1} e S_{i-1}.

Pelo lema 2.4, $D_i \cap \bigcup_{j=1}^{i-1} A_j = \emptyset$. Isso implica que $D_i \subseteq (M_{i-1} \cap S_{i-1})$. Assim, em $M_{i-1} - D_i$, dois terminais estão separados se e só se estão separados em $S_{i-1} - D_i$. Como A_i é uma árvore geradora de $G[\tau_i]$ e T_i é uma árvore de Steiner de (G, τ_i) que não usa vértices de Steiner de S_{i-1}, temos que S_i é uma árvore de Steiner de (G, R) e cada aresta de $M_i \cap S_i$ separa os mesmos pares de terminais em M_i e S_i.

Quanto ao custo de S_i, temos que $c(S_i) = c(S_{i-1}) - c(D_i) + c(T_i) = c(S_{i-1}) - g(M_{i-1}, \tau_i)$ e, por indução, $c(S_i) = c(S_0) - \sum_{j=1}^{i} g(M_{j-1}, \tau_j)$. □

Contraindo-se os vértices de Steiner de S_f que são cópias de um mesmo vértice, obtém-se o S_f devolvido pelo algoritmo das árvores 3-restritas. Tal operação de contração mantém S_f conexo e não altera o custo já que não há custo nas arestas entre os vértices de Steiner duplicados. Podemos concluir então que o S_f devolvido pelo algoritmo contém uma árvore de Steiner de (G, R) e que $c(S_f) = c(S_0) - \sum_{j=1}^{f} g(M_{j-1}, \tau_j)$.

Finalmente temos o teorema mostrando uma razão de aproximação para o algoritmo das árvores 3-restritas.

Teorema 2.6 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. O custo da solução do algoritmo das árvores 3-restritas para (G, c, R) é no máximo $\frac{11}{6}$ do custo de uma árvore de Steiner mínima de (G, c, R).*

Prova. Denote $g(M_{i-1}, \tau_i)$ por g_{τ_i}. Como $\dot{c}_0(M_0) = c(S_0)$, pelos lemas 2.1 e 2.4, temos que,

$$c(S_f) \leq c(S_0) - \sum_{i=0}^{f-1} g_{\tau_i} = c(S_0) - \left(\frac{\dot{c}_f(M_f) - \dot{c}_0(M_0)}{2} \right) \leq \frac{\dot{c}_0(M_0) + \dot{c}_f(M_f)}{2}. \qquad (2.2)$$

Agora mostraremos que $\dot{c}_f(M_f) \leq smt_3(G,c,R)$. Seja T^* uma árvore 3-restrita mínima de (G,c,R) e $\tau_1^*,\ldots,\tau_l^*$ o conjunto de terminais de cada um dos componentes cheios de T^* (note que $2 \leq |\tau_i^*| \leq 3$ para $1 \leq i \leq l$). Como $mst(G[R],\dot{c}[\tau_1^*,\ldots,\tau_l^*]) = 0$, podemos escrever o custo de M_f como

$$\begin{aligned}
\dot{c}_f(M_f) &= mst(G[R],\dot{c}_f) - mst(G[R],\dot{c}_f[\tau_0^*,\ldots,\tau_l^*]) \\
&= \sum_{k=1}^{l}(mst(G[R],\dot{c}_f[\tau_1^*,\ldots,\tau_{k-1}^*]) - mst(G[R],\dot{c}_f[\tau_1^*,\ldots,\tau_k^*])) \\
&\leq \sum_{k=1}^{l}(mst(G[R],\dot{c}_f) - mst(G[R],\dot{c}_f[\tau_k^*])) \qquad (2.3)\\
&= \sum_{k=1}^{l}(g(M_f,\tau_k^*) + smt(G,c,\tau_k^*)) \\
&\leq \sum_{k=1}^{l} smt(G,c,\tau_k^*) = c(T_3^*). \qquad (2.4)
\end{aligned}$$

A desigualdade (2.3) vale por (1.1) e pelo corolário 1.3 e a desigualdade (2.4) vale pela linha 5 do algoritmo que garante que $g(M_f,\tau_k^*) \leq 0$ para todo τ_k^*.

Como $\dot{c}_0(M_0) = smt_2(G,c,R)$ e $\dot{c}_f(M_f) \leq smt_3(G,c,R)$, podemos substituir esses valores na desigualdade (2.2) e dividir pelo custo de uma árvore de Steiner mínima de (G,c,R), ficando com

$$\begin{aligned}
\frac{c(S_f)}{smt(G,c,R)} &\leq \frac{smt_2(G,c,R)}{2smt(G,c,R)} + \frac{smt_3(G,c,R)}{2smt(G,c,R)} \\
&\leq \frac{r_2+r_3}{2} \qquad (2.5)\\
&= \frac{2}{2} + \frac{5/3}{2} \qquad (2.6)\\
&= \frac{11}{6}.
\end{aligned}$$

Onde (2.5) vem de $r_k := \max\left\{\frac{smt_k(G,c,R)}{smt(G,c,R)}\right\}$ e a igualdade (2.6) vale pelo lema 1.6. □

Infelizmente, não existe na literatura uma família de instâncias tal que a aplicação deste algoritmo resulte na razão de aproximação anunciada. Logo existe a possibilidade deste algoritmo ter uma razão de aproximação menor do que $\frac{11}{6}$.

2.3 Algoritmo original de Zelikovsky

O algoritmo originalmente proposto por Zelikovsky diferia em vários pontos da versão apresentada na figura 2.1. Primeiramente, na linha 13, $c_i(p(e))$ era zerado. O algoritmo

não construía iterativamente as árvores $S_0, \ldots, S_f$. Ele apenas guardava em um conjunto W os vértices de Steiner das árvores $\tau_1, \ldots, \tau_f$ e no fim devolvia uma árvore geradora mínima de $(G[R \cup W], c)$. Veja a figura 2.8.

A análise apresentada para a variante pode ser facilmente alterada para o algoritmo original. Basta notar que o lema 2.4 continua válido mesmo com a alteração na linha 13 e que $mst(G[R \cup W_f], c) \leq c(S_f)$.

Algoritmo das árvores 3-restritas(G, c, R)
1 Seja $\dot{c}_0$ a restrição de c a $G[R]$;
2 Seja M_0 uma árvore geradora mínima de $(G[R], \dot{c}_0)$;
3 $W_0 := \emptyset$;
4 $i := 1$;
5 **Enquanto** existir $\tau \subseteq R$, $|\tau| = 3$, com $g(M_{i-1}, \tau) > 0$ **faça**
6 Seja $\tau_i \subseteq R$ com $|\tau_i| = 3$ e $g(M_{i-1}, \tau_i)$ máximo;
7 Seja T_i uma árvore 3-restrita mínima de (G, c, τ_i);
8 $W_i := W_{i-1} \cup VS(T_i)$
9 Seja D_i um conjunto remoção de τ_i em M_{i-1};
10 Seja p a função projeção de D_i em τ_i relativa a M_{i-1};
11 $c_i := c_{i-1}$;
12 **Para** cada e em D_i **faça**
13 $\dot{c}_i(p(e)) := 0$;
14 $A_i := A_i \cup \{p(e)\}$;
15 $M_i := M_{i-1} - D_i + A_i$;
16 $i := i + 1$;
17 $f := i - 1$;
18 Seja T uma árvore geradora mínima de $(G[R \cup W_f], c)$;
19 **Devolva** T.

Figura 2.8: Algoritmo das árvores 3-restritas original

Capítulo 3

Algoritmo de Berman e Ramayer

O algoritmo descrito no capítulo anterior utiliza árvores 3-restritas mínimas para escolher os vértices de Steiner que estarão na solução produzida pelo algoritmo. Uma idéia natural para melhorar esse algoritmo é a utilização de árvores de Steiner mínimas sobre conjuntos de até k terminais, onde k é uma constante maior ou igual a 3.

Berman e Ramayer [6] apresentaram um algoritmo que utiliza esta idéia resultando em uma razão de aproximação menor que $\frac{11}{6}$ para o PSG.

3.1 Descrição do algoritmo

Na verdade, Berman e Ramayer descreveram uma família de algoritmos: um algoritmo para cada $k \geq 3$. A entrada de cada algoritmo é um grafo completo G, uma função c de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e um conjunto R de vértices — os terminais. Como saída, o algoritmo produz uma árvore de Steiner de (G, R) cujo custo dividido pelo custo de uma solução ótima do PSG para (G, c, R) é no máximo

$$r_2 - \sum_{i=3}^{k} \frac{r_{i-1} - r_i}{i-1}.$$

Pelo lema 1.6, para k suficientemente grande a razão de aproximação do algoritmo de Berman e Ramayer se aproxima de 1,734.

Note que quando $k = 3$ a expressão acima coincide com a razão demonstrada para o algoritmo das árvores 3-restritas. Embora neste caso a delimitação na razão de aproximação seja a mesma para os dois algoritmos, eles são diferentes, ou seja, para uma mesma instância, eles podem produzir soluções diferentes. Há no entanto uma forma de modificar o algoritmo de Berman e Ramayer de forma que, para $k = 3$, ele funcione exatamente como o algoritmo das árvores 3-restritas.

O algoritmo de Berman e Ramayer, para um certo k, é dividido em duas fases: a *fase de avaliação* e a *fase de construção*, que passamos a descrever.

Fase de avaliação

Seja $\dot{c}$ a restrição de c a $G[R]$. Esta fase inicia com uma árvore geradora mínima M de $(G[R], \dot{c})$ e pilhas π_j $(3 \leq j \leq k)$ inicialmente vazias. Esta fase processa, em ordem crescente de cardinalidade, todos conjuntos τ de terminais que têm cardinalidade entre 3 e k.

Em cada iteração, esta fase toma um conjunto τ de terminais e constrói uma árvore de Steiner mínima T^*_τ de (G, c, τ), um conjunto remoção D de τ em M e a função projeção p de D em τ relativa a M.

Se o ganho de τ em M é positivo, empilha-se algumas informações para usar na fase de construção, altera-se convenientemente o custo $\dot{c}$ na imagem de p e modifica-se a árvore M, removendo cada aresta e de D e adicionando $p(e)$. A fase termina quando todos os conjuntos são analisados e ela devolve uma árvore geradora mínima de $(G[R], \dot{c})$ e as pilhas π_j para $3 \leq j \leq k$. Abaixo segue a descrição precisa da fase de avaliação.

Algoritmo BR_k**-avaliação**(G, c, R)
1 Seja $\dot{c}$ a restrição de c a $G[R]$;
2 Seja M uma árvore geradora mínima de $(G[R], \dot{c})$;
3 **Para** j de 3 até k **faça**
4 $\quad \pi_j := \emptyset$;
5 $\quad$ **Para** cada $\tau \subseteq R$ tal que $|\tau| = j$ **faça**
6 $\quad\quad$ **Se** $g(M, \tau) > 0$ **então**
7 $\quad\quad\quad$ $ganho := g(M, \tau)$;
8 $\quad\quad\quad$ Seja D um conjunto remoção de τ em M;
9 $\quad\quad\quad$ Seja p a função projeção de D em τ;
10 $\quad\quad\quad$ Empilha $[\tau, D, p, \dot{c}]$ em π_j;
11 $\quad\quad\quad$ **Para** cada e em D **faça**
12 $\quad\quad\quad\quad$ $\dot{c}(p(e)) := \dot{c}(e) - ganho$;
13 $\quad\quad\quad\quad$ $M := M - e + p(e)$;
14 **Devolva** $M, \{\pi_j\}_{j=3}^k$.

Figura 3.1: Fase de avaliação

Sejam M_2 e $\dot{c}_2$ os valores iniciais de M e $\dot{c}$ e, para $3 \leq j \leq k$, sejam M_j e $\dot{c}_j$ os valores de M e $\dot{c}$ após o processamento de todos os conjuntos τ onde $|\tau| = j$, ou seja, ao final da iteração j do **para** da linha 3. Denotando por $Q(j)$ o conjunto dos τ armazenados em π_j e por g_τ o valor de $g(M, \tau)$ na linha 6 da iteração em que τ é processado, temos o seguinte invariante do algoritmo BR_k-avaliação.

Lema 3.1 *Para todo j, $2 \leq j \leq k$, M_j é uma árvore geradora mínima de $(G[R], \dot{c}_j)$.*

Além disso,

$$\dot{c}_j(M_j) = \begin{cases} mst(G[R], \dot{c}_j) & se\ j = 2, \\ \dot{c}_{j-1}(M_{j-1}) - (j-1)\sum_{\tau \in Q(j)} g_\tau & se\ 3 \leq j \leq k. \end{cases}$$

Prova. A prova é por indução em j. Para $j = 2$, o lema é trivial pois M_2 é uma árvore geradora mínima de $(G[R], \dot{c})$ e $\dot{c}_2 = \dot{c}$. Suponha agora que as afirmações do lema valham para j tal que $2 \leq j < k$.

Queremos mostrar que M_{j+1} é uma árvore geradora mínima de $(G[R], \dot{c}_{j+1})$ e que $\dot{c}_{j+1}(M_{j+1}) = \dot{c}_j(M_j) - j\sum_{\tau \in Q(j+1)} g_\tau$.

Por indução M_j é uma árvore geradora mínima de $(G[R], \dot{c}_j)$. Note que M_j é o valor de M e $\dot{c}_j$ é o valor de $\dot{c}$ no início da iteração $j+1$ do **para** da linha 3. Em cada iteração do **para** da linha 11 modificamos $\dot{c}$ e M. Reduzimos o custo de uma aresta $p(e)$ e trocamos a aresta e por $p(e)$ em M. Por causa da redução do custo de $p(e)$ e, segundo as definições de conjunto remoção e função projeção, é fácil ver que no fim do **para** M é uma árvore geradora mínima de $(G[R], \dot{c})$. Portanto, podemos concluir que M_j é uma árvore geradora mínima de $(G[R], \dot{c}_j)$.

Note que, em cada iteração do **para** da linha 11, o custo de M diminui de g_τ. Logo, após o processamento de todas as arestas de D, o custo de M diminui de $|D|g_\tau = (|\tau|-1)g_\tau$. Observe que, entre M_j e M_{j+1}, são processados todos os subconjuntos de R com $j+1$ terminais. Sejam $\tau_1, \ldots, \tau_p$ os subconjuntos de R em $Q(j+1)$, ou seja, aqueles conjuntos cujo ganho é positivo. A diferença entre M_j e M_{j+1} é de $(|\tau_1|-1)g_{\tau_1} + \cdots + (|\tau_p|-1)g_{\tau_p}$. Como $|\tau_1| = \cdots = |\tau_p| = j+1$, temos que,

$$\dot{c}_j(M_j) - \dot{c}_{j+1}(M_{j+1}) = j \sum_{\tau \in Q(j+1)} g_\tau. \ \square$$

Um outro invariante do algoritmo BR_k-avaliação é dado pelo lema abaixo.

Lema 3.2 *Para qualquer subconjunto τ de R tal que $|\tau| \leq j$, temos que $g(M_j, \tau) \leq 0$, onde $3 \leq j \leq k$.*

Prova. Primeiramente, vamos mostrar que, ao fim da iteração em que τ foi processado pelo algoritmo, $g(M, \tau) \leq 0$. Depois vamos mostrar que $g(M, \tau)$ jamais cresce durante o algoritmo.

Seja τ um conjunto com no máximo j terminais e M e $\dot{c}$ no início da iteração em que τ foi processado pelo algoritmo. Se $g(M, \tau) \leq 0$, não há nada a demonstrar. Se $g(M, \tau) > 0$, denote por M' e $\dot{c}'$ os valores de M e $\dot{c}$ ao final desta iteração. Seja A a imagem da função p definida na linha 9 do algoritmo. Como A é uma árvore geradora $G[\tau]$ e as arestas de A estão em M', o único conjunto remoção de τ em M' é A. Lembrando que

$g(M', \tau) = ind(M', \dot{c}', \tau) - smt(G, c, \tau)$ e que $\dot{c}(A) = ind(M', \dot{c}, \tau)$, podemos reescrever o ganho como sendo

$$\begin{aligned} g(M', \tau) &= \dot{c}(A) - smt(G, c, \tau) \\ &= (\dot{c}(D) - (|\tau| - 1)g(M, \tau)) - smt(G, c, \tau) \\ &= (ind(M, \dot{c}, \tau) - (|\tau| - 1)g(M, \tau)) - (ind(M, \dot{c}, \tau) - g(M, \tau)) \\ &= (2 - |\tau|)g(M, \tau), \end{aligned}$$

que é negativo, pois $|\tau| \geq 3$ e $g(M, \tau)$ é positivo.

Para provar que $g(M, \tau)$ jamais cresce durante o algoritmo, basta mostrar que $g(M, \tau) \geq g(M', \tau)$. Note que M e M' são, respectivamente, árvores geradoras mínimas de $(G[R], \dot{c})$ e $(G[R], \dot{c}')$. Além disso, pelas linhas 6 a 13 da fase de avaliação, temos que $M' := M - D + p(D)$ e, para toda aresta xy em D, $\dot{c}'(p(xy)) \leq ind(M, \dot{c}, xy)$, onde D é um conjunto remoção de τ em M e p a função projeção de D em τ. Pelo lema 1.2, temos que $ind(M, \dot{c}, \tau) \geq ind(M', \dot{c}', \tau)$. Agora, utilizando o lema 1.4 concluímos que $g(M, \tau) \geq g(M', \tau)$. Assim, $g(M, \tau)$ jamais cresce durante o algoritmo BR_k-avaliação.

Com isso podemos concluir a prova do lema. Para qualquer conjunto τ com no máximo j terminais temos que $g(M_j, \tau) \leq 0$ no fim da iteração j do algoritmo BR_k-avaliação. □

Fase de construção

Recorde que a fase de avaliação devolve uma árvore geradora M de $G[R]$ e as pilhas $\{\pi_j\}_{j=3}^{k}$.

A fase de construção utiliza as informações armazenadas nas pilhas para construir uma árvore de Steiner de (G, R). A partir de M, as informações nas pilhas são processadas em ordem reversa, isto é, começamos a retirar as informações armazenadas a partir da pilha π_k até π_3.

Inicialmente começamos com a árvore $S := M$. A cada iteração, desempilha-se um conjunto remoção D, sua correspondente função projeção p e um conjunto τ de terminais. Seja A a imagem de p. Restauramos M, removendo A e incluindo D. Alteramos S de forma que seu custo não suba muito e, no fim da iteração, este não contenha nenhuma aresta de A.

Após o processamento de todas as informações das pilhas, o algoritmo devolve S como solução. Veja na figura 3.2 a descrição precisa da fase de construção.

Denote por S_j os valores de S no início do **para** da linha 2 do algoritmo para $j = k, \ldots, 3$.

Seja c^+ uma função custo sobre as arestas de S definida do seguinte modo:

Algoritmo BR_k**-construção**$(G, R, M, \{\pi_j\}_{j=3}^{k})$
1 $S := M$;
2 **Para** j de k até 3 **faça**
3 **Enquanto** $\pi_j \neq \emptyset$ **faça**
4 Desempilha $[\tau, D, p, \dot{c}]$ de π_j;
5 Seja A a imagem de p;
6 $M := M - A + D$;
7 **Se** $A \subseteq E_S$ **então**
8 Seja T_τ^* uma árvore de Steiner mínima de (G, c, τ);
9 $V_S := V_S \cup VS(T_\tau^*)$;
10 $E_S := E_S \setminus A$;
11 **Para** cada aresta e em T_τ^* **faça**
12 **Se** $E_S \cup \{e\}$ não forma circuito **então**
13 $E_S := E_S \cup \{e\}$;
14 **Senão**
15 **Para** cada aresta f em $A \cap E_S$ **faça**
16 Seja e em M tal que $S - f + e$ é conexo e $\dot{c}(e)$ é mínimo;
17 $S := S - f + e$;
18 **Devolva** S.

Figura 3.2: Fase de construção

$$c^+(e) = \begin{cases} \dot{c}(e) & \text{se } e \in E_{G[R]}, \\ \\ c(e) & \text{caso contrário.} \end{cases}$$

Note que na linha 6 o algoritmo está desfazendo as alterações realizadas em M e $\dot{c}$ na fase de avaliação. Assim, considerando que $\dot{c}$ no início da fase de construção é $\dot{c}_k$, os valores de M e $\dot{c}$ no início do **para** da linha 2 são, respectivamente, M_j e $\dot{c}_j$.

O seguinte invariante vale no algoritmo BR_k-construção.

Lema 3.3 *Para todo j, $k \geq j \geq 2$, S_j é uma árvore de Steiner de (G, R) e o seu custo satisfaz o seguinte*

$$c_j^+(S_j) \leq \begin{cases} \dot{c}_k(M_k) & \text{se } j = k, \\ \\ c_{j+1}^+(S_{j+1}) + (j-1)\sum_{\tau \in Q(j+1)} g_\tau & \text{se } k > j \geq 2. \end{cases}$$

Prova. A prova é por indução em j.

Para $j = k$ as afirmações do lema são triviais pois no início do algoritmo BR_k-construção S_k é igual a M_k. Suponha que as afirmações do lema valham para j tal que $k \geq j > 2$.

Vamos provar que S_{j-1} é uma árvore de Steiner de (G, R) e que seu custo satisfaz $c^+_{j-1}(S_{j-1}) \leq c^+_j(S_j) + (j-2)\sum_{\tau \in Q(j)} g_\tau$.

Para isso vamos analisar as alterações realizadas em S e em seu custo no processamento de uma quádrupla $[\tau, D, p, \dot{c}]$ desempilhada.

Seja S' o valor de S na linha 3 do algoritmo. Mostraremos que mesmo após as modificações de S' o grafo resultante, que chamaremos de S'', é uma árvore de Steiner de (G, R).

Temos dois trechos do algoritmo que modificam S' (linhas 8 a 13 e 15 a 17). Mostraremos que ao término dessas modificações S' continua uma árvore de Steiner de (G, R).

Seja T^*_τ uma árvore de Steiner mínima de (G, c, τ). Nas linhas 8 a 13 inserimos os vértices de Steiner de T^*_τ em S' e substituímos as arestas de A em S' por arestas de T^*_τ tomando o cuidado de não deixar formar circuitos. Note que ao fim dessas instruções o grafo resultante é uma árvore.

Nas linhas 15 a 17 removemos cada aresta em $A \cap E'_S$ por uma aresta e em M de tal modo que $S' - f + e$ é conexo. Como S' é uma árvore então $S' - f + e$ também é uma árvore.

Pelas observações acima e como S' é uma árvore de Steiner de (G, R) pode-se concluir que após processarmos uma quádrupla o grafo S'' é uma árvore de Steiner de (G, R).

Agora vamos analisar o que acontece com $c^+(S')$ em uma iteração do **enquanto** na linha 3. Se todas as arestas de A estão em S', substituímos A por "algumas" arestas de T^*_τ. Assim, o novo custo de S' é, no máximo, $c^+(S') - \dot{c}(A) + smt(G, c, \tau)$. Pela definição de ganho temos que

$$\begin{aligned} c^+(S') &\leq c^+(S') - (\dot{c}(D) - (|\tau| - 1)g_\tau) + (\dot{c}(D) - g_\tau) \\ &\leq c^+(S') + (|\tau| - 1)g_\tau - g_\tau \\ &\leq c^+(S') + (|\tau| - 2)g_\tau. \end{aligned}$$

Agora vamos analisar o caso em que nem todas as arestas de A estão em S'. Nas linhas 13 a 15 do algoritmo, substitui-se todas as arestas que estão em A e em S' por arestas de M. Seja $f := uv$ uma dessas arestas, onde $u, v \in \tau$. Pela fase de avaliação, $\dot{c}(f) = ind(M, \dot{c}, uv) - g_\tau$. Seja e a aresta em M escolhida na linha 14. Note que e está no caminho em M entre u e v, logo $\dot{c}(e) \leq ind(M, \dot{c}, uv)$. Portanto $\dot{c}(e) - \dot{c}(f) \leq \dot{c}(e) - ind(M, \dot{c}, uv) \leq g_\tau$. Como $|A \cap E'_S| \leq |\tau| - 2$, então após processarmos todas as arestas, o custo de S' aumenta de no máximo $(|\tau| - 2)g_\tau$.

Portanto, em ambos os casos, o custo de S' aumenta de no máximo $(|\tau| - 2)g_\tau$ para cada quádrupla desempilhada.

Agora estamos prontos para provar que o lema é verdadeiro para $j - 1$. Observe que, entre S_j e S_{j-1}, são processadas todas as quádruplas armazenadas em π_{j-1}.

Logo, a diferença entre os custos de S_{j-1} e S_j é no máximo o somatório dos aumentos no custo de S para cada quádrupla analisada. Lembrando que $Q(j-1)$ é o conjunto de todos os τ armazenados em π_{j-1}, temos que $c^+_{j-1}(S_{j-1}) - c^+_j(S_j) \leq (j-2)\sum_{\tau \in Q(j)} g_\tau$. $\square$

O algoritmo de Berman e Ramayer consiste da chamada dos algoritmos BR_k-avaliação e BR_k-construção. Pelo lema 3.3 é fácil ver que a solução produzida pelo algoritmo é uma árvore de Steiner em (G, R) além disso as duas fases do algoritmo de Berman e Ramayer podem ser implementadas de forma que o consumo de tempo delas seja polinomial.

3.2 Análise do algoritmo de Berman e Ramayer

Sejam m_j e s_j os valores de $\dot{c}_j(M_j)$ e $c^+_j(S_j)$ nos algoritmos BR_k-avaliação e BR_k-construção respectivamente, para $j = 2, \ldots, k$.

Lema 3.4 *Sejam m_j o valor de M_j na fase de avaliação e t_j o valor de uma árvore j-restrita mínima T^*_j de (G, c, R). Então $m_j \leq t_j$ para $j = 2, \ldots, k$.*

Prova. Sejam $\tau_1, \ldots, \tau_l$ o conjunto de terminais de cada um dos l componentes cheios de T^*_j. Podemos escrever o custo m_j como

$$\begin{aligned}
m_j &= mst(G[R], \dot{c}_j) - mst(G[R], \dot{c}_j[\tau_1, \ldots, \tau_l]) \\
&= \sum_{i=1}^{l} (mst(G[R], \dot{c}_j[\tau_1, \ldots, \tau_{i-1}]) - mst(G[R], \dot{c}_j[\tau_1, \ldots, \tau_i])) \\
&\leq \sum_{i=1}^{l} (mst(G[R], \dot{c}_j) - mst(G[R], \dot{c}_j[\tau_i])) \qquad (3.1) \\
&= \sum_{i=1}^{l} (g(M_j, \tau_i) + smt(G, c, \tau_i)) \\
&\leq \sum_{i=1}^{l} smt(G, c, \tau_i) = c(T^*_j) = t_j. \qquad (3.2)
\end{aligned}$$

A desigualdade (3.1) vale pelo corolário 1.3 e pela equação (1.1), e a desigualdade (3.2) vale pelo lema 3.2, que garante que $g(M_j, \tau_i) \leq 0$. $\square$

A seguir mostramos a prova de uma razão de aproximação para o algoritmo.

Teorema 3.5 *O algoritmo de Berman e Ramayer tem uma razão de aproximação de, no máximo,*

$$r_2 - \sum_{i=3}^{k} \frac{r_{i-1} - r_i}{i-1}, \qquad (3.3)$$

onde $r_k := \sup\left\{\frac{smt_k(G,c,R)}{smt(G,c,R)}\right\}$.

Prova. Pelos lemas 3.1 e 3.3, temos que

$$s_{j-1} - s_j \leq (j-2) \sum_{\tau \in Q(j)} g_\tau \quad \text{para } j = k, \ldots, 3 \text{ e}$$
$$m_{j-1} - m_j = (j-1) \sum_{\tau \in Q(j)} g_\tau \quad \text{para } j = 3, \ldots, k.$$

Substituindo a segunda expressão na primeira obtemos que, para $j = 3, \ldots, k$,

$$\begin{aligned} s_{j-1} - s_j &\leq \frac{j-2}{j-1}(m_{j-1} - m_j) \\ &= (m_{j-1} - m_j) - \frac{1}{j-1}(m_{j-1} - m_j). \end{aligned}$$

Somando-se para todo j, obtemos

$$s_2 - s_k \leq m_2 - m_k - \sum_{j=3}^{k} \frac{1}{j-1}(m_{j-1} - m_j),$$

e como $S_k = M_k$, concluímos que

$$\begin{aligned} s_2 &\leq m_2 - \sum_{j=3}^{k} \frac{1}{j-1}(m_{j-1} - m_j) \\ &= \sum_{j=2}^{k-1} m_j \left(\frac{1}{j-1} - \frac{1}{j}\right) + \frac{1}{k-1} m_k. \end{aligned} \tag{3.4}$$

Agora, considere uma árvore j-restrita mínima T_j de (G, c, R) e seja t_j o seu custo. Pelo lema 3.4, $m_j \leq t_j$ para $2 \leq j \leq k$. Então, usando a equação (3.4), deduzimos que

$$s_2 \leq \sum_{j=2}^{k-1} m_j \underbrace{\left(\frac{1}{j-1} - \frac{1}{j}\right)}_{\geq 0} + \frac{1}{k-1} m_k \leq \sum_{j=2}^{k-1} t_j \left(\frac{1}{j-1} - \frac{1}{j}\right) + \frac{1}{k-1} t_k = t_2 - \sum_{j=3}^{k} \frac{t_{j-1} - t_j}{j-1}.$$

Dividindo tudo pelo custo de uma árvore de Steiner mínima de (G, R, c), obtemos

$$\frac{s_2}{smt(G,c,R)} \leq \frac{t_2}{smt(G,c,R)} - \sum_{j=3}^{k} \frac{t_{j-1} - t_j}{smt(G,c,R)(j-1)} = r_2 - \sum_{j=3}^{k} \frac{r_{j-1} - r_j}{j-1}. \quad \square$$

3.3 Comparação com o algoritmo das árvores 3 - restritas

Se, na fase de avaliação do algoritmo de Berman e Ramayer para $k = 3$, escolhemos as triplas em ordem do ganho — as de ganho maior primeiro — então o algoritmo de Berman e Ramayer reduz-se ao algoritmo das árvores 3-restritas.

O lema 2.2 no fundo garante que apenas as triplas escolhidas pelo algoritmo das árvores 3-restritas são empilhadas e que, na fase de construção, a condição da linha 7 está sempre satisfeita, ou seja, uma árvore de Steiner de uma tripla empilhada na fase de avaliação será incluída na árvore final.

Capítulo 4

Algoritmo do ganho relativo

Os algoritmos das árvores 3-restritas e o de Berman e Ramayer utilizam a função ganho para escolher árvores k-restritas mínimas, onde $k = 3$ para o primeiro algoritmo e k é uma constante fixa maior ou igual a 3 para o segundo. Os vértices de Steiner dessas árvores escolhidas compõem, junto com os terminais, uma árvore de Steiner produzida por esses dois algoritmos como solução do PSG.

O algoritmo do ganho relativo, também proposto por Zelikovsky [49], é guloso e utiliza a mesma idéia de escolher árvores k-restritas mínimas para melhorar o custo da solução corrente. A diferença fundamental entre este algoritmo e os anteriores está no processo de escolha das árvores k-restritas.

Novamente, o algoritmo que apresentaremos consiste na verdade em uma família de algoritmos — um algoritmo para cada valor de k. Assim, assumimos no restante do capítulo que k é uma constante fixa maior ou igual a 3.

4.1 Ganho relativo

Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices — os terminais. Considere também a restrição $\dot{c}$ de c a $G[R]$, uma árvore geradora mínima M de $(G[R], \dot{c})$ e um subconjunto τ de R. O *ganho relativo de τ em M*, denotado por $g_r(M, \tau)$, é a razão entre o custo de uma árvore de Steiner mínima de (G, c, τ) e o custo de um conjunto remoção de τ em M. Em fórmula,

$$g_r(M, \tau) := \frac{smt(G, c, \tau)}{ind(M, \dot{c}, \tau)}. \tag{4.1}$$

Note que ficou implícita a dependência do ganho relativo das funções c e $\dot{c}$.

4.2 Descrição do algoritmo

A entrada do algoritmo é um grafo completo G, uma função c de E_G em $Q_{>0}$ sa-

tisfazendo a desigualdade triangular e um conjunto R de vértices de G. Como saída, o algoritmo produz uma árvore de Steiner de (G, R) cujo custo dividido pelo custo de uma solução ótima do PSG para (G, c, R) é no máximo $(1 + \ln(\frac{r_2}{r_k}))r_k$. Quando k cresce, essa delimitação se aproxima de $1 + \ln(r_2) = 1 + \ln(2) < 1{,}694$.

Seja $\dot{c}$ a restrição de c a $G[R]$ e, para um conjunto τ de no máximo k terminais, denote por T_τ uma árvore de Steiner mínima de (G, c, τ).

O algoritmo começa com uma árvore geradora mínima M de $(G[R], \dot{c})$ e um conjunto W de vértices, inicialmente vazio. Em cada iteração, o algoritmo escolhe um conjunto τ e constrói T_τ e um conjunto remoção D de τ em M que minimiza $g_r(M, \tau)$. O algoritmo adiciona os vértices de Steiner de T_τ a W e zera o custo $\dot{c}(e)$ de cada aresta e em $G[R]$ que está na imagem da função projeção de D em τ. Além disso, o algoritmo altera M de forma a mantê-la uma árvore geradora mínima de $(G[R], \dot{c})$. O algoritmo pára quando o custo de M é igual a zero. Neste ponto o algoritmo devolve uma árvore geradora mínima em $(G[R \cup W], c)$. Na figura 4.1 está a descrição precisa do algoritmo do ganho relativo.

Algoritmo do ganho relativo(G, c, R)
1 Seja $\dot{c}_0$ a restrição de c a $G[R]$;
2 Seja M_0 uma árvore geradora mínima de $(G[R], \dot{c}_0)$;
3 $i := 1$;
4 $W_0 := \emptyset$;
5 **Enquanto** $\dot{c}_{i-1}(M_{i-1}) > 0$ **faça**
6 Seja $\tau_i \subseteq R$ com $2 \leq |\tau_i| \leq k$ e $g_r(M_{i-1}, \tau_i)$ mínimo;
7 Seja T_i uma árvore k-restrita mínima de (G, c, τ_i);
8 $W_i := W_{i-1} \cup VS(T_i)$;
9 Seja D_i um conjunto remoção de τ_i em M_{i-1};
10 Seja p a função projeção de D_i em τ_i relativa a M_{i-1};
11 $A_i := \emptyset$;
12 $c_i := c_{i-1}$;
13 **Para** cada e em D_i **faça**
14 $\dot{c}_i(p(e)) := 0$;
15 $A_i := A_i \cup \{p(e)\}$;
16 $M_i := M_{i-1} - D_i + A_i$;
17 $i := i + 1$;
18 $f := i - 1$;
19 Seja T uma árvore geradora mínima de $(G[R \cup W_f], c)$;
20 **Devolva** T.

Figura 4.1: Algoritmo do ganho relativo

Note que

$$g_r(M_{i-1}, \tau_i) := \frac{smt(G, c, \tau_i)}{ind(M_{i-1}, \dot{c}_{i-1}, \tau_i)} \leq 1, \tag{4.2}$$

pois existe τ tal que $g_r(M_{i-1}, \tau) = 1$. Basta tomar como τ os extremos de uma aresta de $M_0 \cap M_{i-1}$. (Se não há nenhuma aresta em $M_0 \cap M_{i-1}$, então o custo de M_{i-1} é nulo e o algoritmo teria parado.) Além disso, note que o algoritmo do ganho relativo pode ser implementado de forma que seu consumo de tempo seja polinomial no tamanho da sua entrada.

4.3 Análise do algoritmo do ganho relativo

O lema a seguir mostra a relação entre a árvore devolvida como solução pelo algoritmo e as árvores de Steiner escolhidas durante o algoritmo.

Lema 4.1 *Seja T a solução produzida pelo algoritmo do ganho relativo e $T_1, \ldots, T_f$ as árvores de Steiner escolhidas. Temos que*

$$c(T) \leq \sum_{j=1}^{f} c(T_j).$$

Prova. Seja $T' := \bigcup_{j=1}^{f} T_j$. Para provar o lema é suficiente mostrar que T' é conexo e que seu conjunto de vértices é igual ao conjunto de vértices de T.

Suponha por absurdo que T' não é conexo. Logo existem pelo menos dois componentes A e B em T'. Como T' é formado pela união de T_j para $j = 1, \ldots, f$ então em cada um dos componentes A e B existe pelo menos um terminal. Seja xy uma aresta de M_0 tal que x está em A e y está em B. Note que não existe nenhum T_j, $j = 1, \ldots, f$, que contenha x e y, logo xy está em M_f e $g_r(M_f, \{x, y\}) = 1$. Portanto $c_f(xy) = c_0(xy) > 0$ contrariando a condição de parada do algoritmo. Assim, podemos concluir que T' é conexo.

A união dos vértices de Steiner das árvores de Steiner escolhidas pelo algoritmo é igual a W_f. Suponha por absurdo que existe um vértice v em R tal que v não está em T'. Note que existe uma aresta uv em M_0 para algum u em R e tal aresta está em M_f já que v não está em nenhum T_j. Além disso $\dot{c}_f(uv) = \dot{c}_0(uv) > 0$, ou seja, $\dot{c}(M_f) > 0$, uma contradição. Logo o conjunto de vértices de T é igual ao conjunto de vértices de T'. □

O próximo lema mostra um delimitador superior da razão entre o custo da solução produzida pelo algoritmo do ganho relativo e do custo de uma árvore k-restrita mínima de (G, c, R).

Lema 4.2 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. O custo da solução do algoritmo*

do ganho relativo sobre o custo de uma árvore k-restrita mínima de (G, c, R) é no máximo $(1 + \ln \frac{r_2}{r_k})$.

Prova. Seja T^* uma árvore k-restrita mínima em (G, c, R) e, para cada componente cheio T_j^* de T^*, considere τ_j^* os vértices terminais de T_j^* para $1 \leq j \leq l$.

Em cada iteração, o algoritmo escolhe um conjunto τ_i de no máximo k terminais que minimiza o ganho relativo. Assim,

$$g_r(M_{i-1}, \tau_i) \leq \min_j \left\{ g_r(M_{i-1}, \tau_j^*) \right\}. \tag{4.3}$$

Sabemos que, para números não-negativos a_j e b_j, $1 \leq j \leq l$,

$$\min_j \frac{a_j}{b_j} \leq \frac{\sum_j a_j}{\sum_j b_j}.$$

Tomando $a_j = c(T_j^*)$ e $b_j = ind(M_{i-1}, \dot{c}_{i-1}, \tau_j^*)$ e, utilizando (4.1) e (4.3), temos que

$$g_r(M_{i-1}, \tau_i) \leq \min_j \left\{ \frac{c(T_j^*)}{ind(M_{i-1}, \dot{c}_{i-1}, \tau_j^*)} \right\} \leq \frac{\sum_j c(T_j^*)}{\sum_j ind(M_{i-1}, \dot{c}_{i-1}, \tau_j^*)}. \tag{4.4}$$

Pelo corolário 1.3, o denominador do lado direito da desigualdade acima pode ser substituído por

$$\sum_{j=1}^{l} ind(M_j', \dot{c}_{i-1}[\tau_1^*, \ldots, \tau_{j-1}^*], \tau_j^*),$$

onde M_j' é uma árvore geradora mínima de $(G[R], \dot{c}_{i-1}[\tau_1^*, \ldots, \tau_{j-1}^*])$ e que, pela definição de índice, é uma soma telescópica. Assim

$$\begin{aligned} g_r(M_{i-1}, \tau_i) &\leq \frac{smt_k(G, c, R)}{mst(G[R], \dot{c}_{i-1}) - mst(G[R], \dot{c}_{i-1}[\tau_1^*, \ldots, \tau_l^*])} \\ &= \frac{smt_k(G, c, R)}{mst(G[R], \dot{c}_{i-1})}, \end{aligned} \tag{4.5}$$

já que $mst(G[R], \dot{c}_{i-1}[\tau_1^*, \ldots, \tau_l^*]) = 0$ pois $\bigcup_{j=1}^{l} T_j^* = T^*$ e conecta R.

Sejam $m_i := mst(G[R], \dot{c}_i)$, $smt_k := smt_k(G, c, R)$ e $smt := smt(G, c, R)$. Usando a definição de ganho relativo, temos que

$$g_r(M_{i-1}, \tau_i) = \frac{c(T_i)}{ind(M_{i-1}, \dot{c}_{i-1}, \tau_i)} = \frac{c(T_i)}{m_{i-1} - m_i}.$$

Portanto, aplicando as desigualdades (4.2) e (4.5),

$$\begin{aligned} \sum_{i=1}^{f} c(T_i) &= \sum_{i=1}^{f} g_r(M_{i-1}, \tau_i)(m_{i-1} - m_i) \\ &\leq \sum_{i=1}^{f} \min \left\{ 1, \frac{smt_k}{m_{i-1}} \right\} (m_{i-1} - m_i). \end{aligned} \tag{4.6}$$

Note que a seqüência $m_0, m_1, \ldots, m_f$, que são os custos das árvores geradoras mínimas em cada iteração do algoritmo, é monotonicamente decrescente, sendo que $m_f = 0$ e $m_0 = mst(G[R], \dot{c})$. Assim, estimamos o somatório em (4.6) por uma integral do seguinte modo

$$\begin{aligned}
\sum_{i=1}^{f} \min\left\{1, \frac{smt_k}{m_{i-1}}\right\}(m_{i-1} - m_i) &\leq \sum_{i=1}^{f} \int_{m_i}^{m_{i-1}} \min\left\{1, \frac{smt_k}{x}\right\} dx \\
&\leq \int_{m_f}^{m_0} \min\left\{1, \frac{smt_k}{x}\right\} dx \\
&= \int_{0}^{smt_k} dx + smt_k \int_{smt_k}^{m_0} \frac{1}{x} dx \\
&= smt_k + smt_k \left(\ln \frac{m_0}{smt_k}\right) \\
&= smt_k \left(1 + \ln \frac{m_0}{smt} \frac{smt}{smt_k}\right) \\
&= smt_k \left(1 + \ln \frac{r_2}{r_k}\right). \qquad (4.7)
\end{aligned}$$

Do lema 4.1 e das desigualdades (4.6) e (4.7), temos que

$$c(T) \leq \sum_{i=1}^{f} c(T_i) \leq (1 + \ln \frac{r_2}{r_k}) smt_k(G, c, R). \ \square$$

Abaixo segue a prova de uma razão de aproximação para o algoritmo do ganho relativo.

Teorema 4.3 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. O custo de uma solução do algoritmo do ganho relativo para (G, c, R) sobre o custo de uma árvore de Steiner mínima de (G, c, R) é no máximo $(1 + \ln \frac{r_2}{r_k}) r_k$.*

Prova. Seja T a solução produzida pelo algoritmo do ganho relativo. Pelo lema 4.2, temos que

$$\frac{c(T)}{smt(G, c, R)} = \frac{c(T)}{smt_k(G, c, R)} \cdot \frac{smt_k(G, c, R)}{smt(G, c, R)} = \frac{c(T)}{smt_k(G, c, R)} r_k \leq \left(1 + \ln \frac{r_2}{r_k}\right) r_k. \ \square$$

Para k suficientemente grande, a delimitação acima se aproxima de $1 + \ln 2 < 1{,}694$.

Similarmente ao algoritmo de Berman e Ramayer, não se conhece uma instância do PSG que resulte exatamente na razão de aproximação demonstrada para o algoritmo do ganho relativo. A instância que resultou na maior razão de aproximação conhecida para o algoritmo do ganho relativo foi exibida por Gröpl, Hougardy, Nierhoff e Prömel [22]. A razão de aproximação obtida foi de 1,33 para k suficientemente grande.

Capítulo 5

Algoritmo do pré-processamento

Com exceção do algoritmo de Berman e Ramayer, os algoritmos das árvores 3-restritas e do ganho relativo são gulosos. Em cada iteração eles escolhem um conjunto τ de no máximo k terminais e uma árvore de Steiner sobre τ que sejam ótimos segundo algum critério de escolha e acrescentam os vértices de Steiner dessa árvore a um conjunto W. Se considerarmos o valor de $smt(G, c, R \cup W)$ durante a execução de cada um desses algoritmos, esse número pode aumentar quando o algoritmo escolhe um vértice de Steiner "errado" para incluir em W.

Karpinski e Zelikovsky [31] apresentam uma maneira de levar em conta esse aumento. O algoritmo proposto por eles utiliza o que chamaremos de *floresta de terminais mínima*. Além disso, o algoritmo do pré-processamento também depende de um parâmetro fixo k que delimita o número de terminais das árvores k-restritas que o algoritmo seleciona.

5.1 Floresta de terminais e ganho ponderado

Sejam G um grafo e R um conjunto de vértices de G — os terminais. Seja T uma árvore de Steiner de (G, R). Lembrando que $VS(T)$ é o conjunto de vértices de Steiner de T, um subgrafo F de T é uma floresta de terminais de (T, R) se, para qualquer v em $VS(T)$, existe um caminho em F que conecta v a algum vértice de R. Veja a figura 5.1.

Uma *floresta de terminais mínima* F^* de (T, c, R) é uma floresta de terminais tal que, para qualquer floresta de terminais F de (T, R), temos que $c(F^*) \leq c(F)$. O custo de F^* é o que chamamos de *perda de* (T, c, R) e denotamos por $perda(T, c, R)$. Note que $perda(T, c, R) = \sum_i perda(T_i, c, \tau_i)$, onde T_i é um componente cheio de T e $\tau_i := VT(T_i)$.

O lema seguinte nos dá um importante limitante superior para a perda de uma árvore de Steiner.

Lema 5.1 *Para qualquer árvore de Steiner T de (G, c, R), cujos vértices de Steiner têm grau pelo menos 3, temos que*

$$perda(T, c, R) \leq \frac{1}{2}\, c(T).$$

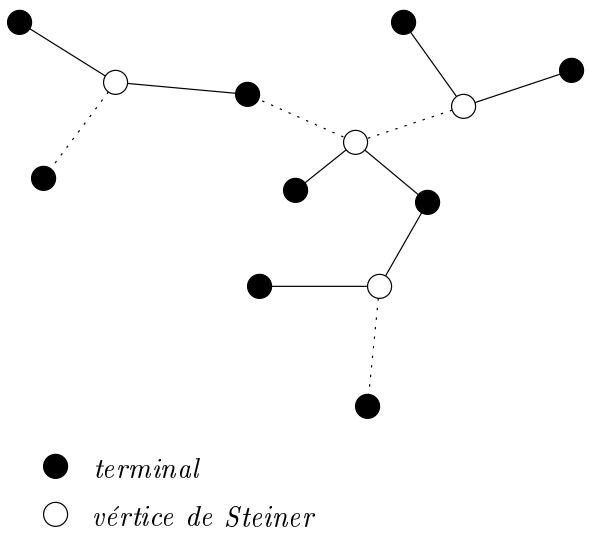

Figura 5.1: Uma árvore de Steiner onde as linhas pontilhadas representam uma floresta de terminais da árvore.

Prova. Para provar este lema, basta mostrar que para qualquer componente cheio T_τ de T vale $perda(T_\tau, c, \tau) \leq \frac{1}{2}\, c(T_\tau)$, onde $\tau = VT(T_\tau)$.

Seja r um terminal de T_τ. Oriente as arestas de T_τ de forma que T_τ torne-se uma árvore enraizada em r.

Agora vamos construir uma floresta de terminais. Para cada vértice de Steiner s, escolha uma das arestas de custo mínimo, digamos f, entrando em s (note que todo vértice de Steiner tem pelo menos uma aresta entrando nele). Seja F o conjunto das arestas escolhidas — F é uma floresta de terminais. De fato, todo vértice de Steiner está conectado em F a uma folha da árvore, que é um terminal. O custo de F é não mais que $\frac{1}{2}\, c(T_\tau)$ pois, para cada aresta e em F, há pelo menos uma aresta de T_τ fora de F de custo pelo menos $c(e)$, já que cada vértice de Steiner em T_τ tem grau pelo menos 3. Veja a figura 5.2. □

Note que toda árvore de Steiner T pode ser transformada em uma árvore de Steiner com custo no máximo $c(T)$ onde cada vértice de Steiner tem grau no mínimo três. A primeira observação é que podemos remover de T os vértices de Steiner de grau zero ou um. Se um vértice de Steiner v tem grau dois, exclua v e as arestas incidentes a v de T e insira a aresta em G que conecta os dois vértices adjacentes a v em T. Por causa da desigualdade triangular, se denotarmos por T' a árvore resultante, temos que $c(T') \leq c(T)$.

Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices — os terminais. Dados uma função $\dot{c}$ de $E_{G[R]}$ em $Q_{>0}$, uma árvore geradora mínima M de $(G[R], \dot{c})$ e um conjunto τ de terminais, o

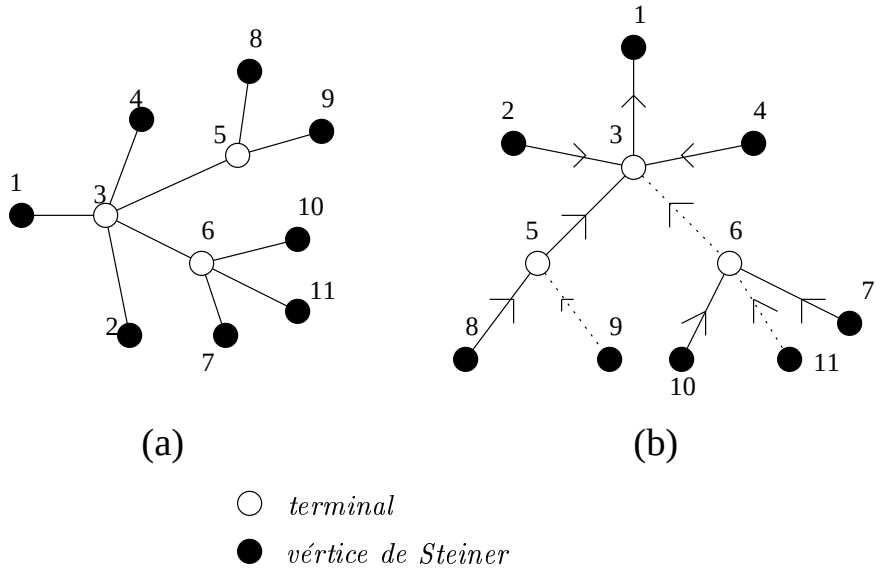

Figura 5.2: (a) Um componente cheio T_τ de T. (b) Uma árvore enraizada T_τ com raiz no vértice 1. Linhas pontilhadas representam arestas da floresta de terminais de T_τ.

ganho ponderado de τ em M, denotado por $g_p(M, \tau)$, é dado por

$$g_p(M, \tau) := \min_{T_\tau} \left\{ \frac{c(T_\tau) + \frac{1}{2} perda(T_\tau, c, \tau)}{ind(M, \dot{c}, \tau)} \right\}, \tag{5.1}$$

onde o mínimo é tomado dentre todas as árvores k-restritas T_τ de (G, τ).

Note que está implícita a dependência do ganho ponderado das funções c e $\dot{c}$.

5.2 Descrição do algoritmo

A entrada do algoritmo é um grafo completo G, uma função c de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e um conjunto R de vértices de G. Como saída, o algoritmo produz uma árvore de Steiner de (G, R) cujo custo dividido pelo custo de uma solução ótima do PSG, para k suficientemente grande, é no máximo 1,645.

Seja $\dot{c}$ a restrição de c a $G[R]$. O algoritmo começa com uma árvore geradora mínima M de $(G[R], \dot{c})$ e um conjunto W de vértices, inicialmente vazio. Em cada iteração, o algoritmo escolhe uma árvore k-restrita de (G, c, τ) com $\tau \subseteq R$ e $|\tau| \leq k$ (note que k está fixo durante o algoritmo) que minimize $g_p(M, \tau)$ e constrói um conjunto remoção D_τ de τ em M. O algoritmo adiciona os vértices de Steiner de T_τ a W e zera o custo $\dot{c}(e)$ de cada aresta e em $G[R]$ que está na imagem da função projeção de D_τ em τ. Além disso, o algoritmo altera M de forma a mantê-la uma árvore geradora mínima de $(G[R], \dot{c})$. O

algoritmo pára quando $\dot{c}(M) = 0$. Neste ponto, o algoritmo executa o algoritmo do ganho relativo para $(G, c, R \cup W)$. Na figura 5.3, está a descrição precisa do algoritmo do pré-processamento.

Algoritmo do pré-processamento(G, c, R)
1 Seja $\dot{c}_0$ a restrição de c a $G[R]$;
2 Seja M_0 uma árvore geradora mínima em $(G[R], \dot{c}_0)$;
3 $i := 1$;
4 **Enquanto** $\dot{c}_{i-1}(M_{i-1}) > 0$ **faça**
5 Seja τ_i tal que $\tau_i \subseteq R$, $2 \leq |\tau_i| \leq k$, e $g_p(M_{i-1}, \tau_i)$ é mínimo;
6 Seja T_i uma árvore k-restrita de (G, τ_i) tal que $g_p(M_{i-1}, \tau_i) = \frac{c(T_i) + \frac{1}{2} perda(T_i, c, \tau_i)}{ind(M_{i-1}, \dot{c}_{i-1}, \tau_i)}$;
7 $W := W \cup VS(T_i)$;
8 Seja D_i um conjunto remoção de τ_i em M_{i-1};
9 Seja p a função projeção de D_i em τ_i relativa a M_{i-1};
10 $A_i := \emptyset$;
11 $\dot{c}_i := \dot{c}_{i-1}$;
12 **Para** cada e em D_i **faça**
13 $\dot{c}_i(p(e)) := 0$;
14 $A_i := A_i \cup \{p(e)\}$;
15 $M_i := M_{i-1} - D_i + A_i$;
16 $i := i + 1$;
17 $f := i - 1$;
18 Seja T' a saída do algoritmo do ganho relativo para $(G, c, R \cup W)$;
19 **Devolva** T'.

Figura 5.3: Algoritmo do pré-processamento

Note que o algoritmo do pré-processamento pode ser implementado de forma que seu consumo de tempo seja polinomial.

Como

$$\frac{c(T_\tau) + \frac{1}{2} perda(T_\tau, c, \tau)}{ind(M_{i-1}, \dot{c}_{i-1}, \tau)} = 1,$$

quando τ é o conjunto dos extremos de uma aresta de M_{i-1} com custo diferente de zero e T_τ é uma árvore k-restrita de (G, τ), temos que

$$g_p(M_{i-1}, \tau_i) = \frac{c(T_i) + \frac{1}{2} perda(T_i, c, \tau_i)}{ind(M_{i-1}, \dot{c}_{i-1}, \tau_i)} \leq 1. \tag{5.2}$$

5.3 Análise do algoritmo do pré-processamento

O algoritmo será analisado em duas fases. A primeira fase compreende as linhas 1 a 17 do algoritmo e a segunda fase corresponde à execução do algoritmo do ganho relativo (linha 18). O seguinte lema corresponde à análise da primeira fase.

Lema 5.2 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. Se $T_1, \ldots, T_f$ são as árvores k-restritas escolhidas pelo algoritmo do pré-processamento para (G, c, R) e $\tau_1, \ldots, \tau_f$ seus respectivos conjuntos de terminais, então*

$$\sum_{i=1}^{f}(c(T_i) + \frac{1}{2}\,perda(T_i, c, \tau_i)) \leq \frac{5}{4} smt_k(G, c, R)\left(1 + \ln \frac{4}{5}\frac{mst(G[R], c)}{smt_k(G, c, R)}\right).$$

Prova. Seja T^* uma árvore k-restrita mínima de (G, c, R) e, para cada componente cheio T_j^* de T^*, considere τ_j^* os terminais de T_j^* para $1 \leq j \leq l$. Por (5.1), temos que

$$g_p(M_{i-1}, \tau_j^*) \leq \frac{c(T_j^*) + \frac{1}{2} perda(T_j^*, c, \tau_j^*)}{ind(M_{i-1}, \dot{c}_{i-1}, \tau_j^*)}.$$

Pela escolha de τ_i na linha 5 do algoritmo, temos que

$$g_p(M_{i-1}, \tau_i) \leq \min_j \left\{ g_p(M_{i-1}, \tau_j^*) \right\} \leq \min_j \left\{ \frac{c(T_j^*) + \frac{1}{2}\, perda(T_j^*, c, \tau_j^*)}{ind(M_{i-1}, \dot{c}_{i-1}, \tau_j^*)} \right\}. \tag{5.3}$$

Sabemos que, para números não-negativos a_j e b_j, $1 \leq j \leq l$,

$$\min_j \frac{a_j}{b_j} \leq \frac{\sum_j a_j}{\sum_j b_j}.$$

Tomando-se $a_j = c(T_j^*) + \frac{1}{2} perda(T_j^*, c, \tau_j^*)$ e $b_j = ind(M_{i-1}, \dot{c}_{i-1}, \tau_j^*)$, por (5.3), temos que

$$g_p(M_{i-1}, \tau_i) \leq \frac{\sum_j (c(T_j^*) + \frac{1}{2}\, perda(T_j^*, c, \tau_j^*))}{\sum_j ind(M_{i-1}, \dot{c}_{i-1}, \tau_j^*)}.$$

Pelo corolário 1.3, o denominador do lado direito da desigualdade acima pode ser substituído. Assim, obtemos

$$g_p(M_{i-1}, \tau_i) \leq \frac{\sum_{j=1}^{l} (c(T_j^*) + \frac{1}{2}\, perda(T_j^*, c, \tau_j^*))}{\sum_{j=1}^{l} ind(M_j', \dot{c}_{i-1}[\tau_1^*, \ldots, \tau_{j-1}^*], \tau_j^*)},$$

onde M_j' é uma árvore geradora mínima de $(G[R], \dot{c}_{i-1}[\tau_1^*, \ldots, \tau_{j-1}^*])$. Pelo lema 5.1 e por que $\sum_{j=1}^{l} c(T_j^*) = c(T^*) = smt_k(G, c, R)$, temos que

$$g_p(M_{i-1}, \tau_i) \leq \frac{smt_k(G, c, R) + \frac{1}{2} \cdot \frac{1}{2} smt_k(G, c, R)}{\sum_{j=1}^{l} ind(M', \dot{c}_{i-1}[\tau_1^*, \ldots, \tau_{j-1}^*], \tau_j^*)}.$$

Pela definição de índice, o denominador do lado direito da desigualdade acima é uma soma telescópica. Então

$$\begin{aligned} g_p(M_{i-1},\tau_i) &\leq \frac{\frac{5}{4}smt_k(G,c,R)}{mst(G[R],\dot{c}_{i-1}) - mst(G[R],\dot{c}_{i-1}[\tau_1^*,\ldots,\tau_l^*])} \\ &= \frac{5}{4}\frac{smt_k(G,c,R)}{mst(G[R],\dot{c}_{i-1})}, \end{aligned} \tag{5.4}$$

já que $mst(G[R],\dot{c}_{i-1}[\tau_1^*,\ldots,\tau_l^*]) = 0$ pois $\bigcup_{j=1}^{l} T_j^* = T^*$ e conecta R.

Para simplificar as fórmulas, sejam $m_i := mst(G[R],\dot{c}_i)$, $perda_i := perda(T_i,c,\tau_i)$, $smt_k := smt_k(G,c,R)$ e $smt := smt(G,c,R)$. Usando a definição de ganho ponderado, temos que

$$g_p(M_{i-1},\tau_i) = \frac{c(T_i)+\frac{1}{2}perda_i}{ind(M_{i-1},\dot{c}_{i-1},\tau_i)} = \frac{c(T_i)+\frac{1}{2}perda_i}{m_{i-1}-m_i}.$$

Portanto, aplicando-se a igualdade acima e as desigualdades (5.2) e (5.4),

$$\begin{aligned} \sum_{i=1}^{f}(c(T_i)+\frac{1}{2}perda_i) &= \sum_{i=1}^{f} g_r(M_{i-1},\tau_i)(m_{i-1}-m_i) \\ &\leq \sum_{i=1}^{f}\min\left\{1,\frac{5}{4}\frac{smt_k}{m_{i-1}}\right\}(m_{i-1}-m_i). \end{aligned} \tag{5.5}$$

Note que a seqüência $m_0, m_1, \ldots, m_f$, que são os custos das árvores geradoras mínimas em cada iteração do algoritmo, é monotonicamente decrescente, sendo que $m_f = 0$ e $m_0 = mst(G[R],\dot{c})$. Assim, podemos limitar o somatório em (5.5) por uma integral do seguinte modo.

$$\begin{aligned} \sum_{i=1}^{f}\min\left\{1,\frac{5}{4}\frac{smt_k}{m_{i-1}}\right\}(m_{i-1}-m_i) &\leq \sum_{i=1}^{f}\int_{m_i}^{m_{i-1}}\min\left\{1,\frac{5}{4}\frac{smt_k}{x}\right\}dx \\ &= \int_{m_f}^{m_0}\min\left\{1,\frac{5}{4}\frac{smt_k}{x}\right\}dx \\ &= \int_0^{\frac{5}{4}smt_k} dx + \frac{5}{4}smt_k\int_{\frac{5}{4}smt_k}^{m_0}\frac{1}{x}dx \\ &= \frac{5}{4}smt_k + \frac{5}{4}smt_k\left(\ln\frac{m_0}{\frac{5}{4}smt_k}\right) \\ &= \frac{5}{4}smt_k\left(1+\ln\frac{4}{5}\frac{m_0}{smt_k}\right), \end{aligned}$$

e com isso concluímos a prova do lema. □

A seguir vamos nos dedicar à análise da segunda fase do algoritmo do pré-processamento. Primeiramente mostraremos um delimitador superior no custo de uma árvore k-restrita mínima de $(G,c,R\cup W)$.

Lema 5.3 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular, R um conjunto de vértices de G e W o conjunto de vértices de G construído na primeira fase do algoritmo do pré-processamento. Se $T_1, \ldots, T_f$ são as árvores k-restritas selecionadas pelo algoritmo do pré-processamento para (G, c, R) e $\tau_1, \ldots, \tau_f$ seus respectivos conjuntos de terminais, então*

$$smt_k(G, c, R \cup W) \leq smt_k(G, c, R) + \sum_{i=1}^{f} perda(T_i, c, \tau_i).$$

Prova. Seja T^* uma árvore k-restrita mínima de (G, c, R) e F_i uma floresta de terminais mínima de (T_i, c, τ_i) para $1 \leq i \leq f$. Para provar o lema, basta mostrar que o grafo $T^* + F_1 + \cdots + F_f$ contém uma árvore k-restrita de $(G, R \cup W)$.

Primeiro note que $T^* + F_1 + \cdots + F_f$ é conexo, pois $\bigcup_{i=1}^{f} \tau_i = R$, todo componente conexo de F_i contém pelo menos um terminal de τ_i e T^* conecta todos os terminais. Assim, seja T' uma árvore geradora de $T^* + F_1 + \cdots + F_f$. Basta mostrar que T' é uma árvore k-restrita de $(G, R \cup W)$.

Note que toda aresta de F_i que está em T' é um componente cheio de T', pois todo vértice de F_i é terminal em T'. (Para todo $1 \leq i \leq f$, temos que $VS(T_i) \subseteq W$.) Os demais componentes cheios de T' são subárvores de componentes cheios de T^* e portanto têm no máximo o mesmo número de folhas que o correspondente componente cheio de T^*, ou seja, são também k-árvores. Logo T'_k é uma árvore k-restrita de $(G, R \cup W)$ e de fato $T^* + F_1 + \cdots + F_f$ contém uma árvore k-restrita de $(G, R \cup W)$. Como $smt_k(G, c, R \cup W) \leq c(T') \leq smt_k(G, c, R) + \sum_{i=1}^{f} perda(T_i, c, \tau_i)$, o lema segue. □

O teorema abaixo mostra uma razão de aproximação para o algoritmo do pré-processamento.

Teorema 5.4 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. A razão entre o custo da solução do algoritmo do pré-processamento e o custo de uma árvore de Steiner mínima de (G, c, R) é menor que 1,645 para k suficientemente grande.*

Prova. Considere o conjunto W de vértices de Steiner construído pelo algoritmo e sejam $T_1, \ldots, T_f$ as árvores k-restritas mínimas selecionadas pelo algoritmo para (G, c, R) e $\tau_1, \ldots, \tau_f$ seus respectivos conjuntos de terminais. Para simplicar, denotemos $mst(G[R], c)$ por mst, $smt(G, c, R)$ por smt, $smt_k(G, c, R)$ por smt_k e $smt_k(G, c, R \cup W)$ por smt'_k.

Pelo lema 4.2 do algoritmo do ganho relativo, temos que

$$c(T') \leq \left(1 + \ln \frac{mst}{smt'_k}\right) smt'_k \leq \left(1 + \ln \frac{\sum_{i=1}^{f} c(T_i)}{smt'_k}\right) smt'_k, \tag{5.6}$$

pois $mst \leq \sum_{i=1}^{f} c(T_i)$.

Seja

$$B_k := \frac{5}{4} smt_k \left(1 + \ln \frac{4}{5} \frac{mst}{smt_k}\right).$$

Pelo lema 5.2, temos que

$$\sum_{i=1}^{f} c(T_i) \leq B_k - \frac{1}{2} \sum_{i=1}^{f} perda(T_i, c, \tau_i) = \frac{1}{2}(2B_k - \sum_{i=1}^{f} perda(T_i, c, \tau_i))$$

e, combinando o lema 5.3, concluímos que

$$\sum_{i=1}^{f} c(T_i) \leq \frac{1}{2}(2B_k + smt_k - smt'_k). \tag{5.7}$$

Aplicando-se (5.7) em (5.6), deduzimos que

$$\begin{aligned} c(T') &\leq \left(1 + \ln \frac{2B_k + smt_k - smt'_k}{2smt'_k}\right) smt'_k \\ &= f\left(\frac{a_k - smt'_k}{2}\right), \end{aligned} \tag{5.8}$$

onde $a_k := 2B_k + smt_k$ e

$$f(x) = \left(1 + \ln \frac{x}{a_k - 2x}\right)(a_k - 2x).$$

Mas o máximo da função $f(x)$ é atingido quando a sua derivada é nula, ou seja, quando

$$\frac{a_k - 2x}{2x} = \ln \frac{x}{a_k - 2x}. \tag{5.9}$$

Definindo $y = \frac{a_k - 2x}{2x}$, a equação acima corresponde a $y = \ln \frac{1}{2y}$, cuja solução é $y_0 := 0{,}3515\ldots$. Portanto, como a solução de (5.9) é $x_0 := \frac{a_k}{2(1+y_0)}$, então o máximo da função $f(x)$ é

$$f(x_0) = \left(1 + \ln \frac{x_0}{2x_0\, y_0}\right) 2x_0\, y_0 = (1 + y_0) 2y_0 \frac{a_k}{2(1 + y_0)} = a_k\, y_0 = (2B_k + smt_k) y_0,$$

pois $a_k - 2x_0 = 2x_0\, y_0$ e $y_0 = \ln \frac{1}{2y_0}$.

Assim, de (5.8), concluímos que

$$c(T') \leq f(x_0) = (2B_k + smt_k) y_0,$$

e o limite da razão de aproximação do algoritmo do pré-processamento quando k tende a infinito é

$$\begin{aligned}
\lim_{k\to\infty} \frac{c(T')}{smt} &\leq \lim_{k\to\infty} \frac{(smt_k + 2B_k)\, y_0}{smt} \\
&= y_0 \left(\lim_{k\to\infty} \frac{smt_k}{smt} + 2 \lim_{k\to\infty} \frac{B_k}{smt} \right) \\
&\leq y_0 \left(1 + \frac{5}{2} \left(1 + \ln \frac{8}{5} \right) \right) \\
&\leq 1{,}6443\ldots .
\end{aligned}$$

Portanto, para k suficientemente grande, temos que

$$\frac{c(T')}{smt} < 1{,}645. \quad \square$$

A escolha de 1/2 como fator multiplicativo da perda na definição do ganho ponderado resulta na melhor razão de aproximação obtida por esta análise e, como nos algoritmos anteriores, não se conhece instância do PSG que, quando executado o algoritmo do pré-processamento, resulte na razão de aproximação demonstrada.

Hougardy e Prömel [27] generalizaram o algoritmo do pré-processamento. Ao invés de duas fases, o algoritmo proposto por eles é composto por n fases. Basicamente, em cada uma delas o algoritmo escolhe algumas árvores k-restritas e adiciona seus vértices de Steiner ao conjunto de terminais formando um novo conjunto de terminais para a fase seguinte processar. Eles mostraram que este algoritmo atinge uma razão de aproximação de no máximo 1,598 para n e k suficientemente grandes.

Capítulo 6

Algoritmo de Robins e Zelikovsky

Este algoritmo, projetado por Robins e Zelikovsky [42], é o melhor algoritmo de aproximação conhecido para o PSG até o momento. Como os algoritmos apresentados nos capítulos anteriores, ele também utiliza árvores k-restritas, onde k é uma constante fixa maior ou igual a 3, para produzir como solução uma árvore de Steiner.

A novidade deste algoritmo em relação aos anteriores está na utilização do *ganho relativo à perda* e da *contração de florestas de terminais*. A contração de florestas de terminais de (T_τ, τ) em contraposição à contração de T_τ possibilita a escolha, em uma iteração futura, de uma árvore k-restrita que contenha mais de um terminal de uma árvore k-restrita já escolhida. Com exceção do algoritmo de Berman e Ramayer, essa possibilidade não existe nos algoritmos anteriores.

6.1 Contração de florestas de terminais e ganho relativo à perda

Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices — os terminais. Considere também uma função $\dot{c}$ de $E_{G[R]}$ em $Q_{>0}$, um conjunto de pelo menos dois terminais τ, uma árvore de Steiner T_τ de (G, τ) e uma floresta de terminais mínima F de (T_τ, c, τ).

Para cada aresta e de $T_\tau - F$ seja $q(e)$ a aresta de $G[R]$ cujos extremos são os terminais de cada um dos componentes de F conectados pela aresta e. Se $|\tau| = 2$ então $e = q(e)$. A *contração de F em $\dot{c}$* é a função que denotamos por $\dot{c}[T_\tau, F]$ de $E_{G[R]}$ em $Q_{>0}$, definida por

$$\dot{c}[T_\tau, F](f) = \begin{cases} \dot{c}(f) & \text{se } f \neq q(e) \text{ para todo } e \text{ em } E_{T_\tau - F}, \\ \min\{\dot{c}(f), c(e)\} & \text{se } f = q(e) \text{ para algum } e \text{ em } E_{T_\tau - F}. \end{cases}$$

Para simplificar, denotamos $\dot{c}[T_1, F_1][T_2, F_2] \cdots [T_i, F_i]$ por $\dot{c}[T_1 \ldots T_i, F_1 \ldots F_i]$.

A função q definida nas arestas de $T_\tau - F$ é denominada de *projeção da contração de F em T_τ relativa a $G[R]$*. Denotamos por $Q(T_\tau, F)$ a imagem da função q. Veja na figura 6.1 um exemplo de contração de florestas de terminais. Seja T o componente cheio da figura. As linhas grossas representam as arestas de uma floresta de terminais mínima F de T. Para cada aresta e (linha normal) de $T - F$, o custo da aresta $q(e)$ (linha pontilhada) é alterado para o $\min\{c(e), \dot{c}(q(e))\}$.

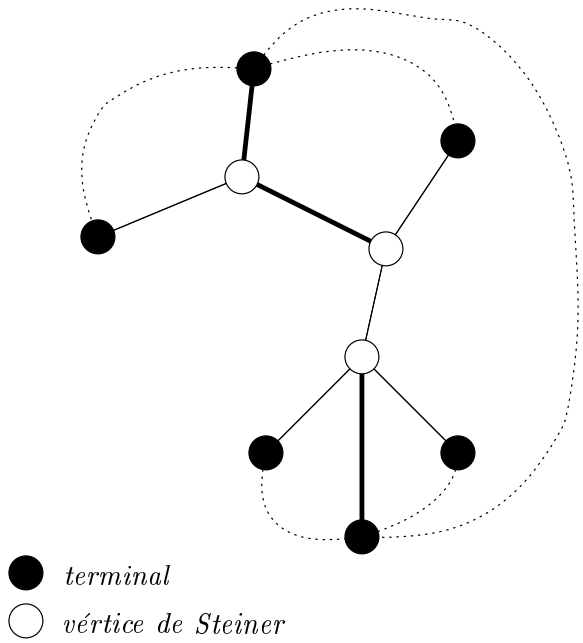

Figura 6.1: Exemplo de contração de uma floresta de terminais.

Nosso objetivo, através do lema a seguir, é mostrar uma desigualdade envolvendo floresta de terminais e contração de florestas de terminais muito semelhante a uma propriedade de conjunto remoção provada no lema 1.2.

Lema 6.1 *Sejam G um grafo, c uma função de E_G em $Q_{>0}$, R um conjunto de vértices de G, $\dot{c}$ a restrição de c a $G[R]$, τ_1 um conjunto de vértices de R, T_1 uma árvore de Steiner de (G, τ_1) e F_1 uma floresta de terminais mínima de (T_1, c, τ_1). Se $\dot{c}'$ é uma função em $E_{G[R]}$ que coincide com $\dot{c}$ exceto em uma aresta e de $G[R]$ onde $\dot{c}'(e) \leq \dot{c}(e)$, então*

$$mst(G[R], \dot{c}) - mst(G[R], \dot{c}[T_1, F_1]) \geq mst(G[R], \dot{c}') - mst(G[R], \dot{c}'[T_1, F_1]).$$

Prova. Sejam $e_1, \ldots, e_j$ as arestas de $T_1 - F_1$ e q a projeção da contração de F_1 em T_1 relativa a $G[R]$. Seja M_0 uma árvore geradora mínima de $(G[R], \dot{c})$ e $\dot{c}_0 = \dot{c}$. Para

$i = 1, \ldots, j$, seja $\dot{c}_i$ a função custo definida da seguinte maneira

$$\dot{c}_i(f) = \begin{cases} \dot{c}_{i-1}(f) & \text{se } f \neq q(e_i), \\ \\ \min\{\dot{c}_{i-1}(f), c(e_i)\} & \text{se } f = q(e_i) \end{cases}$$

e seja M_i uma árvore geradora mínima de $(G[R], \dot{c}_i)$. Note que

$$\dot{c}_{i-1}(M_{i-1}) - \dot{c}_i(M_i) = \max\{0, ind(M_{i-1}, \dot{c}_{i-1}, q(e_i)) - c(e_i)\}.$$

Portanto, como $\dot{c}_j = \dot{c}[T_1, F_1]$, temos que

$$mst(G[R], \dot{c}) - mst(G[R], \dot{c}[T_1, F_1]) = \sum_{i=1}^{j} \max\{0, ind(M_{i-1}, \dot{c}_{i-1}, q(e_i)) - c(e_i)\}. \quad (6.1)$$

Agora, seja $\dot{c}'_0 = \dot{c}'$ e $\dot{c}'_i$ uma função custo definida do seguinte modo

$$\dot{c}'_i(f) = \begin{cases} \dot{c}_i(f) & \text{se } f \neq e, \\ \\ \min\{\dot{c}_i(f), \dot{c}'(e)\} & \text{se } f = e. \end{cases}$$

Seja

$$M'_i := \begin{cases} M_i & \text{se } \dot{c}'_i(e) \geq ind(M_i, \dot{c}_i, e), \\ \\ M_i - e' + e & \text{se } \dot{c}'_i(e) < ind(M_i, \dot{c}_i, e) \text{ e } e' \text{ é o índice de } e \text{ em } M_i. \end{cases}$$

Note que M'_i é uma árvore geradora mínima de $(G[R], \dot{c}'_i)$. Para $i = 1, \ldots, j$ temos, pelo lema 1.2, que

$$ind(M_{i-1}, \dot{c}_{i-1}, e_i) \geq ind(M'_{i-1}, \dot{c}'_{i-1}, e_i). \quad (6.2)$$

Além disso,

$$\dot{c}'_{i-1}(M'_{i-1}) - \dot{c}'_i(M'_i) = \max\{0, ind(M'_{i-1}, \dot{c}'_{i-1}, q(e_i)) - c(e_i)\}.$$

Portanto, como $\dot{c}'_j = \dot{c}'[T_1, F_1]$, temos que

$$mst(G[R], \dot{c}') - mst(G[R], \dot{c}'[T_1, F_1]) = \sum_{i=1}^{j} \max\{0, ind(M'_{i-1}, \dot{c}'_{i-1}, q(e_i)) - c(e_i)\}. \quad (6.3)$$

Por (6.1), (6.2) e (6.3) podemos concluir que

$$mst(G[R], \dot{c}) - mst(G[R], \dot{c}[T_1, F_1]) \geq mst(G[R], \dot{c}') - mst(G[R], \dot{c}'[T_1, F_1]). \quad \square$$

Corolário 6.2 *Sejam G um grafo, c uma função de E_G em $Q_{>0}$, R um conjunto de vértices de G e $\dot{c}$ a restrição de c a $G[R]$. Respectivamente, sejam τ_0 e τ_1 conjuntos não-vazios de vértices de R, T_0 e T_1 árvores de Steiner de (G, τ_0) e (G, τ_1), F_0 e F_1 florestas de terminais mínima de (T_0, c, τ_0) e (T_1, c, τ_1). Então*

$$mst(G[R], \dot{c}) - mst(G[R], \dot{c}[T_1, F_1]) \geq mst(G[R], \dot{c}[T_0, F_0]) - mst(G[R], \dot{c}[T_0, F_0][T_1, F_1]).$$

Prova. Basta aplicar sucessivamente o lema 6.1 com cada aresta de $Q[T_0, F_0]$ no papel de e. □

O *ganho relativo à perda de τ em $\dot{c}$*, denotado por $g_l(\tau, \dot{c})$, é dado por

$$g_l(\tau, \dot{c}) := \min_{T_\tau, F} \left\{ \frac{perda(T_\tau, c, \tau)}{mst(G[R], \dot{c}) - mst(G[R], \dot{c}[T_\tau, F])} \right\},$$

onde o mínimo é tomado dentre todas as árvores k-restritas T_τ de (G, τ) e todas as florestas de terminais mínimas F de (T_τ, c, τ). Se o denominador for nulo, considere $g_l(\tau, \dot{c}) := +\infty$.

Note que o denominador do ganho relativo à perda é uma variante do custo de um conjunto remoção, que utiliza a contração de florestas de terminais no lugar da função redução.

6.2 Descrição do algoritmo

O algoritmo depende de um parâmetro fixo k. A entrada do algoritmo é um grafo completo G, uma função c de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e um conjunto R de vértices de G. Como saída, o algoritmo produz uma árvore de Steiner de (G, R) cujo custo dividido pelo custo de uma solução ótima do PSG para (G, c, R), para k suficientemente grande, é no máximo 1,55.

O algoritmo começa com uma função $\dot{c}$ que inicialmente é a restrição de c a $G[R]$ e um conjunto W de vértices inicialmente vazio. Em cada iteração, o algoritmo escolhe $\tau \subseteq R$ com $3 \leq |\tau| \leq k$, uma árvore k-restrita T_τ de (G, τ) e uma floresta de terminais F de (T_τ, τ) que minimiza $g_l(\tau, \dot{c})$. O algoritmo armazena os vértices de Steiner de T_τ em W e começa uma nova iteração com a contração de F em $\dot{c}$ no lugar de $\dot{c}$.

O algoritmo pára quando não existe τ tal que $3 \leq |\tau| \leq k$ e $g_l(\tau, \dot{c}) < 1$. Neste ponto, o algoritmo constrói uma árvore geradora mínima de $(G[R \cup W], c)$ como saída do algoritmo. Na figura 6.2, está a descrição precisa do algoritmo de Robins e Zelikovsky.

Note que o algoritmo de Robins e Zelikovsky pode ser implementado de forma que seu consumo de tempo seja polinomial.

É importante reforçar que em uma iteração, ao escolhermos uma árvore k-restrita, existe a possibilidade de que pares de terminais desta árvore k-restrita possam ser compartilhados por uma outra árvore k-restrita escolhida futuramente pelo algoritmo. Assim, por causa da contração de florestas de terminais, o algoritmo está descartando menos árvores k-restritas do que se fosse utilizada a operação de redução.

Algoritmo de Robins e Zelikovsky(G, c, R)

1 Seja $\dot{c}_0$ a restrição de c a $G[R]$;
2 $W_0 := \emptyset$;
3 $i := 1$;
4 **Enquanto** existir $\tau \subseteq R$ com $3 \leq |\tau| \leq k$ tal que $g_l(\tau, \dot{c}_{i-1}) < 1$ **faça**
5 Seja $\tau_i \subseteq R$ com $3 \leq |\tau_i| \leq k$ e $g_l(\tau_i, \dot{c}_{i-1})$ mínimo;
6 Sejam T_i uma árvore k-restrita de (G, c, τ_i)
e F_i uma floresta de terminais mínima de (T_i, c, τ_i) para as quais
$g_l(\tau_i, \dot{c}_{i-1}) = \frac{perda(T_i, c_{i-1}, \tau_i)}{mst(G[R], \dot{c}_{i-1}) - mst(G[R], \dot{c}_{i-1}[F_i, \tau_i])}$;
7 $W_i := W_{i-1} \cup VS(T_i)$;
8 $\dot{c}_i := \dot{c}_{i-1}[T_i, F_i]$;
9 $i := i + 1$;
10 $f := i - 1$;
11 Seja T uma árvore geradora mínima de $(G[R \cup W_f], c)$;
12 **Devolva** T.

Figura 6.2: Algoritmo de Robins e Zelikovsky

6.3 Análise do algoritmo de Robins e Zelikovsky

O lema abaixo mostra um limitante inferior para o custo de uma árvore k-restrita mínima que será muito útil na demonstração de uma razão de aproximação do algoritmo.

Lema 6.3 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. Seja T^* uma árvore k-restrita mínima arbitrária de (G, c, R) e t o número de componentes cheios de T^*. Para $j = 1, \ldots, t$ considere as triplas (T_j^*, τ_j^*, F_j^*) onde T_j^* é um componente cheio de T^*, τ_j^* é o conjunto dos terminais de T_j^* e F_j^* é uma floresta de terminais mínima de (T_j^*, c, τ_j^*). Então*

$$smt_k(G, c, R) = c(T^*) \geq mst(G[R], \dot{c}[T_1^* \ldots T_t^*, F_1^* \ldots F_t^*]) + perda(T^*, c, R).$$

Prova. Seja $Q(T_j^*, F_j^*)$ a imagem da função projeção da contração de T_j^* em F_j^* relativa a $G[R]$. Note que este conjunto é um conjunto minimal de arestas que conecta τ_j^*, ou seja, $Q(T_j^*, F_j^*)$ é uma árvore geradora de $G[\tau_j^*]$. Como T_j^* são os componentes cheios de T^*, podemos concluir que $\bigcup_{j=1}^{t} Q(T_j^*, F_j^*)$ são as arestas de uma árvore geradora M de $G[R]$ pois $\bigcup_{j=1}^{t} T_j^* = T^*$. De fato, um circuito em M induziria um circuito em T.

Denotando $\dot{c}[T_1^* \ldots T_t^*, F_1^* \ldots F_t^*]$ por $\dot{c}_*$, vale que

$$\begin{aligned} smt_k(G, c, R) &= c(T^*) \\ &= \sum_{j=1}^{t} c(T_j^*) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{j=1}^{t}(c(T_j^* - F_j^*) + c(F_j^*)) \\
&= \sum_{j=1}^{t}\dot{c}_*(Q(T_j^*, F_j^*)) + \sum_{j=1}^{t} perda(T_j^*, c, \tau_j^*) \\
&= \dot{c}_*(M) + perda(T^*, c, R) \\
&\geq mst(G[R], \dot{c}_*) + perda(T^*, c, R),
\end{aligned}
$$

concluindo a prova do lema. □

Baseando-se no lema 6.3 podemos definir um delimitador superior para o custo de uma solução produzida pelo algoritmo.

Lema 6.4 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. No algoritmo de Robins e Zelikovsky, temos que*

$$mst(G[R \cup W_f], c) \leq mst(G[R], \dot{c}_f) + \sum_{j=1}^{f} perda(T_j, c, \tau_j).$$

Prova. Seja M uma árvore geradora mínima de $G([R], \dot{c}_f)$. Considere o subgrafo H de $G[R \cup W_f]$ obtido de M pela substituição de cada aresta e em $M \cap \bigcup_{j=1}^{f} Q(T_j, F_j)$ por uma aresta f em $T_j - F_j$ tal que $\dot{c}_f(e) = c(f)$ e pela floresta de terminais F_j para algum j. Assim,

$$c(H) \leq \dot{c}_f(M) + \sum_{j=1}^{f} c(F_j).$$

Além disso, H é um subgrafo gerador conexo de $G[R \cup W_f]$. De fato, para cada substituição, $F_j + f$ conecta os extremos da aresta e removida. Portanto

$$mst(G[R \cup W_f], c) \leq c(H) \leq mst(G[R], \dot{c}_f) + \sum_{j=1}^{f} c(F_j). \quad \square$$

Lema 6.5 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. Sejam $\dot{c}$ a restrição de c a $G[R]$, T^* uma árvore k-restrita mínima arbitrária de (G, c, R) e t o número de componentes cheios de T^*. Para $j = 1, \ldots, t$, considere as triplas (T_j^*, τ_j^*, F_j^*) onde T_j^* é um componente cheio de T^*, τ_j^* é o conjunto dos terminais de T_j^* e F_j^* é uma floresta de terminais mínimas de (T_j^*, c, τ_j^*). No algoritmo de Robins e Zelikovsky temos que*

$$mst(G[R], \dot{c}_f) \leq mst(G[R], \dot{c}[T_1^* \ldots T_t^*, F_1^* \ldots F_t^*]) + perda(T^*, c, R).$$

Prova. Denote $\dot{c}_f[T_1^* \ldots T_j^*, F_1^* \ldots F_j^*]$ por c_f^j e $\dot{c}[T_1^* \ldots T_j^*, F_1^* \ldots F_j^*]$ por c_0^j. Pela definição de contração de florestas de terminais mínimas, temos que

$$mst(G[R], c_0^t) \geq mst(G[R], c_f^t).$$

Pela desigualdade acima e pelo corolario 6.2, vale que

$$\begin{aligned} mst(G[R], \dot{c}_f) - mst(G[R], c^t) &\leq mst(G[R], c_f^0) - mst(G[R], c_f^t) \\ &= \sum_{j=1}^{t} (mst(G[R], c_f^{j-1}) - mst(G[R], c_f^j)) \\ &\leq \sum_{j=1}^{t} (mst(G[R], \dot{c}_f) - mst(G[R], \dot{c}_f[T_j^*, F_j^*])). \quad (6.4) \end{aligned}$$

A condição da linha 4 do algoritmo de Robins e Zelikovsky garante que $g_l(\tau_j^*, \dot{c}_f) \geq 1$, ou seja, que

$$mst(G[R], \dot{c}_f) - mst(G[R], \dot{c}_f[T_j^*, F_j^*]) \leq perda(T_j^*, c, \tau_j^*).$$

Combinando a desigualdade acima com (6.4) obtemos que

$$mst(G[R], \dot{c}_f) - mst(G[R], c_0^t) \leq \sum_{j=1}^{t} perda(T_j^*, c, \tau_j^*) = perda(T^*, c, R). \quad \square$$

Finalmente, o próximo lema dá uma delimitação superior para o custo da solução produzida pelo algoritmo. Sua prova se assemelha muito à prova do lema 5.2.

Lema 6.6 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. Seja T^* uma árvore k-restrita mínima arbitrária de (G, c, R) e l o número de componentes cheios de T^*. O custo da solução produzida pelo algoritmo de Robins e Zelikovsky é delimitado por*

$$mst(G[R \cup W_f], c) \leq \begin{cases} smt_k(G, c, R) \quad \text{se } perda(T^*, c, R) = 0, \\ \\ smt_k(G, c, R) + perda(T^*, c, R) \cdot \ln\left(1 + \frac{mst(G[R], c) - smt_k(G, c, R)}{perda(T^*, c, R)}\right) \\ \qquad \text{se } perda(T^*, c, R) > 0. \end{cases}$$

Prova. Considere a tripla (T_j^*, τ_j^*, F_j^*), para $j = 1, \ldots, t$, tal que T_j^* é um componente cheio de T^*, τ_j^* é o conjunto dos terminais de T_j^* e F_j^* é uma floresta de terminais mínima de (T_j^*, c, τ_j^*).

Se $perda(T^*, c, R) = 0$, então T^* é uma árvore 2-restrita mínima de (G, c, R), ou seja, T^* é uma árvore geradora mínima de $(G[R], c)$ que, nesse caso, coincide com uma árvore de Steiner mínima de (G, c, R), isto é, o algoritmo devolve uma solução ótima.

Supondo que $perda(T^*, c, R) > 0$, em cada iteração o algoritmo escolhe um conjunto τ_i de terminais que minimiza o ganho relativo à perda. Assim,

$$g_l(\tau_i, \dot{c}_{i-1}) \leq \min_j \left\{ g_l(\tau_j^*, \dot{c}_{i-1}) \right\}. \tag{6.5}$$

Sabemos que, para números não-negativos a_j e b_j, $1 \leq j \leq t$,

$$\min_j \frac{a_j}{b_j} \leq \frac{\sum_j a_j}{\sum_j b_j}.$$

Tomando $a_j = perda(T_j^*, c, \tau_j^*)$ e $b_j = mst(G[R], \dot{c}_{i-1}) - mst(G[R], \dot{c}_{i-1}[T_j^*, F_j^*])$ e, utilizando a definição do ganho relativo à perda, temos que

$$\begin{aligned} g_l(\tau_i, \dot{c}_{i-1}) &\leq \min_j \left\{ \frac{perda(T_j^*, c, \tau_j^*)}{mst(G[R], \dot{c}_{i-1}) - mst(G[R], \dot{c}_{i-1}[T_j^*, F_j^*])} \right\} \\ &\leq \frac{\sum_j perda(T_j^*, c, \tau_j^*)}{\sum_j (mst(G[R], \dot{c}_{i-1}) - mst(G[R], \dot{c}_{i-1}[T_j^*, F_j^*]))}. \end{aligned}$$

Pelo corolário 6.2, o denominador acima pode ser substituído por

$$\sum_{j=1}^{t} (mst(G[R], \dot{c}_{i-1}[T_1^* \ldots T_{j-1}^*, F_1^* \ldots F_{j-1}^*]) - mst(G[R], \dot{c}_{i-1}[T_1^* \ldots T_j^*, F_1^* \ldots F_j^*])),$$

que é uma soma telescópica. Assim obtemos que

$$\begin{aligned} g_l(\tau_i, \dot{c}_{i-1}) &\leq \frac{perda(T^*, c, R)}{mst(G[R], \dot{c}_{i-1}) - mst(G[R], \dot{c}_{i-1}[T_1^* \ldots T_t^*, F_1^* \ldots F_t^*])} \\ &\leq \frac{perda(T^*, c, R)}{mst(G[R], \dot{c}_{i-1}) - mst(G[R], \dot{c}[T_1^* \ldots T_t^*, F_1^* \ldots F_t^*])}, \end{aligned} \tag{6.6}$$

já que, pela definição de contração de florestas de terminais, é fácil ver que $mst(G[R], \dot{c}_{i-1}[T_1^* \ldots T_t^*, F_1^* \ldots F_t^*]) \leq mst(G[R], \dot{c}[T_1^* \ldots T_t^*, F_1^* \ldots F_t^*])$.

Para simplificar as fórmulas, sejam $m^* := mst(G[R], \dot{c}[T_1^* \ldots T_t^*, F_1^* \ldots F_t^*])$, $perda_i := perda(T_i, c, \tau_i)$, $perda^* := perda(T^*, c, R)$, $m_i := mst(G[R], \dot{c}_i)$ e $smt_k := smt_k(G, c, R)$. Pela escolha de τ_i, T_i e F_i, temos que

$$\sum_{i=1}^{f} perda_i = \sum_{i=1}^{f} g_l(\tau_i, \dot{c}_{i-1})(m_{i-1} - m_i).$$

Substituindo na expressão acima $g_l(\tau_i, \dot{c}_{i-1})$ pela melhor delimitação entre as dadas por (6.6) e pela condição de parada do algoritmo, temos que

$$\sum_{i=1}^{f} perda_i \leq \sum_{i=1}^{f} \min \left\{ 1, \frac{perda^*}{m_{i-1} - m^*} \right\} (m_{i-1} - m_i).$$

Note que $m_0 = mst(G[R], \dot{c}_0)$ e $m_f = mst(G[R], \dot{c}_f)$. Assim,

$$\begin{aligned}
\sum_{i=1}^{f} perda_i &\leq \int_{m_f - m^*}^{m_0 - m^*} \min\left\{1, \frac{perda^*}{x}\right\} dx \\
&\leq \int_{m_f - m^*}^{perda^*} dx + perda^* \int_{perda^*}^{m_0 - m^*} \frac{dx}{x} \qquad (6.7) \\
&= perda^* + m^* - m_f + perda^* \cdot \ln\left(\frac{m_0 - m^*}{perda^*}\right) \\
&\leq smt_k - m_f + perda^* \cdot \ln\left(1 + \frac{m_0 - smt_k}{perda^*}\right). \qquad (6.8)
\end{aligned}$$

Primeiramente observe em (6.7) que $m_0 - m^* \geq perda^* \geq m_f - m^*$. De fato, $m_0 \geq smt_k \geq perda^* + m^*$ pelo lema 6.3 e que $m_f \leq m^* + perda^*$ pelo lema 6.5. Portanto a quebra da integral é válida. A desigualdade (6.8) vale pois a derivada da função

$$f(x) = x + perda^* \cdot \ln\left(\frac{m_0 - x}{perda^*}\right)$$

é não-negativa para $x \leq smt_k - perda^*$ e portanto $f(m^*) \leq f(smt_k - perda^*)$ já que $m^* \leq smt_k - perda^*$ pelo lema 6.3.

Pelo lema 6.4 sabemos que $mst(G[R \cup W_f], c) \leq m_f + \sum_{i=1}^{f} perda_i$. Logo

$$mst(G[R \cup W_f], c) \leq smt_k + perda^* \cdot \ln\left(1 + \frac{m_0 - smt_k}{perda^*}\right),$$

e com isso concluímos a prova do lema. □

O teorema abaixo mostra uma razão de aproximação para o algoritmo de Robins e Zelikovsky.

Teorema 6.7 *Sejam G um grafo completo, c uma função de E_G em $Q_{>0}$ satisfazendo a desigualdade triangular e R um conjunto de vértices de G. A razão entre o custo da solução produzida pelo algoritmo de Robins e Zelikovsky e o custo de uma árvore de Steiner mínima de (G, c, R) é, no máximo,*

$$r_k\left(1 + \frac{\ln 3}{2}\right).$$

Prova. Para simplificar, denotemos $mst(G[R], c)$ por mst, $smt(G, c, R)$ por smt, $smt_k(G, c, R)$ por smt_k e $perda(T^*, c, R)$ por $perda^*$, onde T^* é uma árvore k-restrita mínima arbitrária de (G, c, R). Se $perda^* = 0$, o teorema vale trivialmente pelo lema 6.6. Podemos supor que $perda^* > 0$. Lembre-se que $mst \leq 2\, smt \leq 2\, smt_k$. Se T

é uma árvore de Steiner de (G, R) devolvida pelo algoritmo de Robins e Zelikovsky, o lema 6.6 e a desigualdade anterior garante que

$$c(T) \leq smt_k \left(1 + \frac{perda^*}{smt_k} \cdot \ln\left(1 + \frac{smt_k}{perda^*}\right)\right). \tag{6.9}$$

Pelo lema 5.1 temos que $perda^* \leq \frac{1}{2} smt_k$. Como $perda^* > 0$ e $smt_k > 0$ podemos concluir que $0 < \frac{perda^*}{smt_k} \leq \frac{1}{2}$. A função $f(x) = x \cdot \ln(1 + \frac{1}{x})$ tem um máximo no intervalo $[0, 1/2]$ quando $x = \frac{1}{2}$. Portanto o máximo de (6.9) é alcançado quando $\frac{perda^*}{smt_k} = \frac{1}{2}$. Assim

$$\frac{c(T)}{smt} \leq \frac{smt_k}{smt}\left(1 + \frac{1}{2} \cdot \ln 3\right) = r_k \left(1 + \frac{\ln 3}{2}\right),$$

que conclui a demonstração do teorema. □

Para k suficientemente grande, a delimitação acima se aproxima de $1 + \frac{\ln 3}{2} < 1{,}55$.

Gröpl, Hougardy, Nierhoff e Prömel [22] mostraram uma instância onde a razão de aproximação do algoritmo de Robins e Zelikovsky quando aplicado a essa instância é de 1,2 para k suficientemente grande. Não se conhece outra instância do PSG que resulte em uma razão de aproximação maior quando aplicado ao algoritmo de Robins e Zelikovsky.

Considerações finais

Os algoritmos que apresentamos são simples se comparados com as suas demonstrações de uma razão de aproximação. Na medida do possível, tentamos estabelecer um padrão para a apresentação dos algoritmos e de suas demonstrações. Como dissemos na introdução deste trabalho, o foco principal no estudo desses algoritmos é a demonstração de uma razão de aproximação e não na determinação precisa do tempo de execução dos algoritmos. A tabela abaixo resume, em ordem cronológica, as principais inovações de um algoritmo para o outro e as respectivas razões de aproximação de cada um deles.

Algoritmo		Características	Razão
MST	1968	solução intuitiva	2
árvores 3-restritas	1990	árvores 3-restritas	1,834
Berman e Ramayer	1991	árvores k-restritas	1,734
ganho relativo	1995	função ganho relativo	1,694
pré-processamento	1996	florestas de terminais e ganho ponderado	1,644
Hougardy e Prömel	1999	generalização do pré-processamento	1,598
Robins e Zelikovsky	2000	contração de florestas de terminais	1,55

Razão de aproximação dos algoritmos apresentados.

Embora as dúvidas iniciais sobre esses algoritmos de aproximação baseados em árvores k-restritas para o PSG já não existam, no decorrer da elaboração deste trabalho foram surgindo outras que gostaríamos de partilhar com os leitores.

A partir do algoritmo das árvores 3-restritas, não existe comprovação que a razão de aproximação demonstrada é a razão real do algoritmo. Assim pode ser possível melhorar a análise ou pode ser possível que exista uma instância que resulte na razão de aproximação demonstrada.

Será que podemos aplicar os algoritmos para o PSG em variantes do problema encontrando, para estes, razões de aproximação melhores do que as conhecidas na literatura? Um exemplo concreto de problema onde árvores k-restritas podem trazer razões de aproximações melhores que a melhor razão conhecida até o momento é o Problema de Steiner com Penalidades (veja detalhes em [20] e no capítulo 4 de [26]).

Será que a idéia de fazer um pré-processamento nos terminais utilizando como base o algoritmo de Robins e Zelikovsky resulta em uma razão de aproximação menor do que 1,55?

Uma outra questão interessante é como os algoritmos descritos nesta dissertação se comportam na prática. Ou seja, quando executados em instâncias construídas aleatoriamente ou obtidas de problemas reais, qual deles obtém a melhor árvore de Steiner? Qual deles consome menos tempo?

Deixamos estas perguntas para leitores interessados em prosseguir no estudo de aproximações para o Problema de Steiner em Grafos e nas suas variantes.

Referências Bibliográficas

[1] Y. P. Aneja, *An integer linear programming approach to the Steiner Problem in Graphs*, Networks 10 (1980), 167-178.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan e M. Szegedy, *Proof verification and hardness of approximation problems*, Proceedings of 33rd Annual Symposium on Foundations of Computer Science (1992), 14-23.

[3] J. E. Beasley, *An algorithm for the Steiner problems in graphs*, Networks 14 (1984), 147-159.

[4] J. E. Beasley, *An SST algorithm for the Steiner problems in graphs*, Networks 19 (1989), 1-16.

[5] P. Berman e M. Karpinski, *On some tighter inapproximability results*, Eletronic Colloquium on Computational Complexity, TR98-065 (1998).

[6] P. Berman e V. Ramayer, *Improved approximations for the Steiner tree problem*, Journal of Algorithms 17 (1994), 381-408.

[7] M. Bern e P. Plassmann, *The Steiner problems with lengths 1 and 2*, Information Processing Letters 32 (1989) 171-176.

[8] J. A. Bondy e U. S. R. Murty, *Graph theory with applications*, Macmillan, Londres (1976).

[9] A. Borchers e D. Du, *The k-Steiner ratio in graphs*, SIAM Journal of Computing 26 (1997), 857-869.

[10] T. H. Cormen, C. E. Rivest e R. L. Rivest, *Introduction to algorithms*, MIT-Press (1998).

[11] R. Courant e H. Robbins, *What's mathematics?*, Oxford University Press (1977).

[12] R. Diestel, *Graph Theory*, Springer-Verlag, New York (1997).

[13] E. W. Dijkstra, *A note on two problems in connection with graphs*, Numerical Math. 1 (1959), 269-271.

[14] S. E. Dreyfus e R. A. Wagner, *The Steiner problem in graphs*, Networks 1 (1971), 195-207.

[15] C. E. Ferreira, *O problema de Steiner em grafos: uma abordagem poliédrica*, Dissertação de Mestrado, IME-USP (1989).

[16] C. E. Ferreira, C. G. Fernandes, F. K. Miyazawa, J. A. R. Soares, J. C. Pina Jr.,K. S. Guimarães, M. H. Carvalho, M. R. Cerioli, P. Feofiloff, R. Dahab e Y. Wakabayashi, *Uma introdução sucinta a algoritmos de aproximação*, Livro do XXIII Colóquio Brasileiro de Matemática, IMPA (2001).

[17] M. R. Garey e D. S. Johnson, *Computers and intractability: A guide to the theory of $\mathcal{NP}$-completeness*, Freeman, San Francisco (1979).

[18] M. R. Garey, R. L. Graham e D. S. Johnson, *The complexity of computing Steiner minimal trees*, SIAM Journal of Applied Mathematics 32 (1977), 835-859.

[19] E. N. Gilbert e H. O. Pollack, *Steiner minimal trees*, SIAM Journal of Applied Mathematics 16 (1968) 1-29.

[20] M. X. Goemans e D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, Journal of the ACM 42 (1995), 1115-1145.

[21] C. Gröpl, S. Hougardy, T. Nierhoff e H. J. Prömel, *Aproximation algorithms for the Steiner tree problem in graphs*, Technical report, Humboldt-Universität zu Berlim (2001).

[22] C. Gröpl, S. Hougardy, T. Nierhoff e H. J. Prömel, *Lower bounds for the approximation algorithms for the Steiner tree problem*, Technical report, Humboldt-Universität zu Berlim (2001).

[23] M. Grötschel e C. L. Monma, *Integer polyhedra arising from certain network design problems with connectivity constraints*, Report No. 104, Universität Augsburg (1988).

[24] S. L. Hakimi, *Steiner´s problem in graphs and its implications*, Networks 1 (1971), 113-133.

[25] J. Håstad, *Some optimal inapproximability results*, Proceedings of the 29^{th} Annual ACM Symposium on the Theory of Computing (1997), 1-10.

[26] D. S. Hochbaum, *Approximation Algorithms for $\mathcal{NP}$-Hard Problems*, PWS Publishing (1997).

[27] S. Hougardy e H. J. Prömel, *A 1.598 approximation algorithm for the Steiner problem in graphs*, Proceedings of the 10^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (1999), 448-453.

[28] F. K. Hwang, D. S. Richards e P. Winter, *The Steiner tree problem*, North-Holland, (1992).

[29] A. B. Kahng e G. Robins, *On optimal interconnections for VLSI*, Kluwer Publishers, (1995).

[30] R. M. Karp, *Reducibility among combinatorial problems*, Miller e Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, Nova York (1972), 85-103.

[31] M. Karpinski e A. Z. Zelikovsky, *New approximation algorithms for the Steiner tree problems*, Journal of Combinatorial Optimization 1 (1997), 47-65.

[32] L. Kou, G. Markowsky e L. Berman, *A fast algorithm for Steiner trees*, Acta Informatica 15 (1981), 141-145.

[33] J. B. Kruskal, *On the shortest spanning tree of a graph and the travelling salesman problem*, Proceedings Amererican Mathematical Socciety 7 (1956), 48-50.

[34] E. L. Lawler, *Combinatorial optimization: networks and matroids*, Holt, Rinehart and Winston, New York (1976).

[35] N. Maculan, *The Steiner problem in graphs*, Annals of Discrete Math. 31 (1987).

[36] K. Mehlhorn, *A faster approximation algorithm for the Steiner problem in graphs*, Information Processing Letters 27 (1988), 125-128.

[37] M. Minoux, *Mathematical programming theory and algorithms*, Jonh Wiley and Sons, New York (1986).

[38] C. H. Papadimitriou e K. Steiglitz, *Combinatorial optimization*, Prentice Hall, Englewood Cliffs, New Jersey (1982).

[39] H. J. Prim, *Shortest connection networks and some generalizations*, Bell System Technical Journal 36 (1957), 1389-1401.

[40] R. C. Prömel e A. Steger, *A new approximation algorithm for the Steiner tree problem with performance ratio* $\frac{5}{3}$, Journal of Algorithms 36 (2000), 89-101.

[41] H. Takahashi e A. Matsuyama, *An approximate solution for the Steiner problem in graphs*, Math. Japonica 24 (1980), 573-577.

[42] G. Robins e A. Z. Zelikovsky, *Improved Steiner tree approximation in graphs*, Proceedings of the 11^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (2000), 770-779.

[43] M. L. Shore, L. R. Foulds e P. B. Gibbons, *An algorithm for the Steiner problem in graphs*, Networks 12 (1982), 323-333.

[44] M. Thimm, *On the approximability of the Steiner tree problem*, Proceedings of the 26^{th} International Symposium on Mathematical Foundations of Computes Science (2001), Springer LNCS.

[45] V .V. Vazirani, *Approximation algorithms*, Springer-Verlag, New York (2001).

[46] D. P. Williamson, M. X. Goemans, M. Mihail e V. V. Vazirani, *A primal-dual approximation algorithm for generalized Steiner network problem*, Combinatorica 15 (1995), 435-454.

[47] P. Winter, *Steiner problem in networks: a survey*, Networks 17 (1987), 129-167.

[48] A. Z. Zelikovsky, *An $\frac{11}{6}$-aproximation algorithm for the network Steiner problem*, Algorithmica 9 (1993), 463-470.

[49] A. Z. Zelikovsky, *Better approximation bounds for the network and Euclidean Steiner tree problems*, Technical report CS-96-06, University of Virginia (1996).

Índice Remissivo