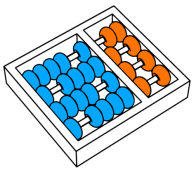
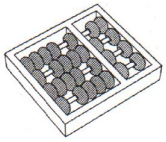


Márcio Félix Reis

“O problema da árvore geradora com muitas folhas.”

CAMPINAS
2014



Universidade Estadual de Campinas
Instituto de Computação

Márcio Félix Reis

"O problema da árvore geradora com muitas folhas."

Orientador(a): **Prof. Dr. Orlando Lee**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

Este exemplar corresponde à versão da Dissertação apresentada à banca examinadora por Márcio Félix Reis, sob orientação de Prof. Dr. Orlando Lee.

Orlando Lee

Assinatura do Orientador(a)

CAMPINAS
2014

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

R277p

Reis, Márcio Félix, 1986-
O problema da árvore geradora com muitas folhas / Márcio Félix Reis. – Campinas, SP : [s.n.], 2014.

Orientador: Orlando Lee.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Algoritmos aproximados. 2. Algoritmos em grafos. 3. Otimização combinatória. I. Lee, Orlando,1969-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: The maximum leaf spanning tree problem
Palavras-chave em inglês:
Approximation algorithm
Graph algorithms
Combinatorial optimization
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Orlando Lee [Orientador]
José Coelho de Pina Junior
Fábio Luiz Usberti
Data de defesa: 08-05-2014
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Defesa de Dissertação de Mestrado em Ciência da Computação, apresentada pelo(a) Mestrando(a) **Márcio Félix Reis**, aprovado(a) em **08 de maio de 2014**, pela Banca examinadora composta pelos Professores Doutores:

Prof(a). Dr(a). José Coelho de Pina Junior
Titular

Prof(a). Dr(a). Fábio Luiz Usberti
Titular

Prof(a). Dr(a). Orlando Lee
Presidente

Instituto de Computação
Universidade Estadual de Campinas

O problema da árvore geradora com muitas folhas.

Márcio Félix Reis[1]

08 de maio de 2014

Banca Examinadora:

- Prof. Dr. Orlando Lee (Orientador)
- Prof. Dr. Fábio Luiz Usberti
  Instituto de Computação - UNICAMP
- Prof. Dr. José Coelho de Pina
  Instituto de Matemática e Estatística - USP
- Prof. Dr. Eduardo Candido Xavier
  Instituto de Computação - UNICAMP (Suplente)
- Prof[a]. Dr[a]. Cristina Gomes Fernandes
  Instituto de Matemática e Estatística - USP (Suplente)

[1]Auxílio financeiro de: CAPES 2011–2013

Resumo

Neste trabalho estudamos o problema da árvore geradora com muitas folhas (PAGMF). Este problema pode ser usado como abstração para diversos problemas práticos e sabe-se que é NP-difícil. Estudamos, implementamos e executamos testes para algoritmos aproximados e exatos para o PAGMF e para um caso particular que considera apenas grafos cúbicos. O principal objetivo do trabalho foi verificar o comportamento prático dos algoritmos estudados. Para as instâncias testadas, em geral, o algoritmo guloso apresentou melhores resultados para o PAGMF e o algoritmo 2-aproximado teve os melhores resultados para os grafos cúbicos.

Abstract

In this work we study the maximum leaf spanning tree problem (MLSTP). This problem can be used as an abstraction for many practical problems and is known to be NP-hard. We studied, implemented and executed tests for approximate and exact algorithms for the MLSTP and for a particular case that considers only cubic graphs. The main objective of this study was to investigate the practical behavior of the algorithms studied. For the tested instances, in general, the greedy algorithm showed better results for the MLSTP and the 2-approximate algorithm had the best results for cubic graphs.

Agradecimentos

Gostaria de agradecer meu orientador Orlando Lee pela oportunidade, pelos ensinamentos, pela paciência e por toda ajuda prestada durante todo o trabalho.

Agradeço aos meus pais, Jesuíno e Rosa, minha irmã, Poliana, e minha noiva, Maríllia, pelo apoio emocional e financeiro que foram fundamentais para que eu conseguisse concluir este trabalho.

Agradeço a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro.

Agradeço ao professor Luidi Simonetti pelos casos de teste fornecidos.

Finalmente, agradeço a todos que me ajudaram de alguma forma, em especial, os alunos do LOCo, meus amigos Divino, Loos, Mario e Murilo, e os professores Alexandre e Clarimar da PUC-GO.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Dado um grafo conexo G, o problema da árvore geradora com muitas folhas (PAGMF) é encontrar uma árvore geradora de G com número máximo de folhas. O PAGMF é NP-difícil [13] e APX-difícil [12].

Uma das maneiras de lidarmos com problemas difíceis é usar algoritmos de aproximação, que são algoritmos que produzem uma solução viável com uma certa garantia de qualidade. Diversos algoritmos aproximados para o PAGMF foram desenvolvidos e neste trabalho fizemos um estudo teórico de alguns deles. Os fatores de aproximação dos algoritmos estudados aqui são determinados por uma análise de pior caso.

Frequentemente o comportamento empírico de um algoritmo é melhor do que sugerido pela análise de pior caso. Um exemplo clássico é o algoritmo para programação linear simplex que apesar de ter complexidade exponencial é muito utilizado porque na prática são raros os casos que tomam essa complexidade. Neste trabalho investigamos o comportamento na prática, em relação à qualidade das soluções geradas, de algoritmos aproximados para o PAGMF e um caso particular.

O PAGMF tem aplicações em áreas como projeto de sistemas distribuídos, *layout* de circuitos e redes de sensores sem fio. Por exemplo, considere um grafo modelando uma rede de computadores, onde os vértices correspondem a computadores e as arestas as conexões entre computadores que permitem a comunicação direta de um com o outro. O objetivo é projetar uma rede em forma de árvore. Considere que os computadores não-folhas têm a capacidade de processamento reduzida pelo trabalho de roteamento de mensagens. Neste caso, a solução do PAGMF poderia fornecer uma solução razoável.

1.1 Organização do Texto

Este trabalho está organizado da seguinte forma. No Capítulo 2 apresentamos definições e notações básicas que serão utilizadas no decorrer do trabalho. Apresentamos conceitos

básicos de otimização combinatória e algoritmos aproximados. Prosseguimos introduzindo alguns conceitos e notações de grafos. Finalmente apresentamos alguns casos particulares para o PAGMF e um problema relacionado.

Nos Capítulos 3 e 4 apresentamos algoritmos aproximados para o caso geral do PAGMF e para grafos cúbicos respectivamente. Apresentamos uma análise da complexidade dos algoritmos e de suas razões de aproximação.

No Capítulo 5 apresentamos uma formulação para o PAGMF usando Programação Linear Inteira e descrevemos os experimentos feitos, detalhes de implementação e os resultados obtidos. A formulação apresentada foi proposta por Lucena, Maculan e Simonetti em [20] e obteve os melhores resultados em média comparadas com duas outras formulações descritas no mesmo artigo. Além disso apresentamos dois métodos de separação para um conjunto exponencial de desigualdades válidas para o modelo.

Finalmente, no Capítulo 6 concluímos o trabalho com uma discussão dos resultados obtidos e sugestões de trabalhos futuros.

Capítulo 2

Preliminares

Neste Capítulo apresentamos definições e notações básicas que serão utilizadas no decorrer do trabalho. Definimos conceitos básicos de otimização combinatória e algoritmos aproximados, onde a principal referência é o livro de Carvalho et al. [4]. Introduzimos também alguns conceitos e notações de grafos. Por fim, apresentamos alguns casos particulares para o PAGMF e um problema relacionado.

2.1 Otimização Combinatória

Um *problema de otimização* consiste de: um conjunto de *instâncias*, um conjunto finito $Sol(I)$ de *soluções viáveis* para cada instância I, e uma função que atribui um número $val(S)$ a cada solução viável S. O número $val(S)$ é o *valor* de S. Para cada instância I de um dado problema associamos um número natural $\langle I \rangle$ que representa o *tamanho da instância*.

Em um problema *de minimização* estamos interessado em encontrar uma solução viável S^* tal que $val(S^*)$ seja mínimo, enquanto em um problema *de maximização* queremos encontrar uma solução viável S^* tal que $val(S^*)$ seja máximo. Neste caso chamamos S^* de *solução ótima*. Denotamos por $opt(I)$ o valor de uma solução ótima, isto é, $opt(I) = val(S^*)$ para alguma solução ótima S^* do problema.

Como o conjunto $Sol(I)$ é finito, um algoritmo possível para resolver um determinado problema de otimização é verificar o valor de val para cada solução viável da instância na busca do melhor. Mas, em geral, existe uma quantidade exponencial em $\langle I \rangle$ de soluções viáveis e portanto teríamos um algoritmo inviável na prática. Para alguns problemas de otimização combinatória, algoritmos polinomiais são conhecidos, mas para muitos outros não.

2.2 Algoritmos de Aproximação

Entendemos por algoritmo eficiente um algoritmo cujo consumo de tempo é polinomial em função do tamanho da entrada. Segundo a teoria da complexidade, não existe um algoritmo eficiente que resolva um problema NP-difícil a menos que $P = NP$. Ao lidar com um problema NP-difícil temos que sacrificar alguma das características: garantia de encontrar uma solução ótima, garantia de tempo de execução, ou a capacidade de resolver instâncias grandes.

Para um problema de otimização combinatória, algoritmos de aproximação são algoritmos que produzem uma solução viável em tempo polinomial e com uma certa garantia de qualidade. Essa não é necessariamente uma solução ótima para o problema. Os algoritmos de aproximação são diferentes de heurísticas, pois garantem encontrar uma solução cujo valor tem uma relação pré-estabelecida com o valor ótimo.

Considere um problema de otimização em que $val(S) \geq 0$ para toda solução viável S. Seja A um algoritmo que devolve uma solução viável $A(I)$ de I para toda instância viável I do problema. Dizemos que A é α-*aproximado* ou que é uma α-*aproximação* para o problema se para toda instância I

$$val(A(I)) \leq \alpha opt(I)$$

se o problema é de minimização e

$$val(A(I)) \geq \alpha opt(I)$$

se o problema é de maximização. O fator α é um número que pode depender de I e é chamado de *razão de aproximação*. No caso de problemas de minimização temos $\alpha \geq 1$. Para problemas de maximização temos $0 < \alpha \leq 1$.

Um *esquema de aproximação* para um problema de otimização é um algoritmo A que recebe um número racional positivo ε e uma instância I e devolve uma solução viável $A(\varepsilon, I)$ com um erro de no máximo ε, ou seja,

$$val(A(\varepsilon, I)) \leq \alpha(1 + \varepsilon)opt(I)$$

se o problema é de minimização e

$$val(A(\varepsilon, I)) \geq \alpha(1 - \varepsilon)opt(I)$$

se o problema é de maximização. Assim podemos definir a taxa máxima de erro do algoritmo escolhendo ε. Dizemos que A é um *esquema de aproximação polinomial (PTAS)* se o algoritmo é polinomial para todo ε fixo. A classe de problemas *NPO* é uma extensão da classe NP para problemas de otimização. A classe APX é formada pelos problemas em NPO para os quais existe uma α-aproximação polinomial para alguma constante α.

Uma *AP-redução* de um problema de otimização Q a um problema de otimização Q' é um terno (f, g, β) onde f e g são algoritmos que executam em tempo polinomial e β é um número racional positivo tais que: f recebe um racional positivo δ e uma instância I de Q, e devolve uma instância $f(\delta, I)$ de Q'; g recebe um racional positivo δ, instância I e um elemento S' em sol$(f(\delta, I))$, e devolve $g(\delta, I, S')$ em sol(I); e para toda instância I de Q, todo número racional positivo δ, e todo S' em sol$(f(\delta, I))$, vale que se $(1-\delta)OPT(f(\delta, I)) \leq val(f(\delta, I), S') \leq (1+\delta)OPT(f(\delta, I))$ então $(1-\beta\delta)OPT(I) \leq val(I, g(\delta, I, S')) \leq (1+\beta\delta)OPT(I)$.

Um problema Q em NPO é *APX-difícil* se todo problema em APX pode ser AP-reduzido a Q. Considerando que $P \neq NP$, dizer que um problema é APX-difícil é o mesmo que dizer que não existe um esquema de aproximação polinomial para esse problema. Para maiores detalhes sobre as classes de complexidade APX e APX-difícil veja [4].

2.3 Grafos

Nesta seção apresentamos alguns conceitos básicos da teoria de grafos. Supomos que o leitor tem relativa familiariadade com esses conceitos, de modo que, nosso objetivo principal é estabelecer a notação que será usada. A terminologia adotada, com algumas exceções, é a mesma usada por Bondy e Murty [1].

Um *grafo* $G = (V, E)$ consiste de um conjunto não vazio V de elementos chamados vértices e um conjunto E de arestas. Uma *aresta* é um par não ordenado de elementos distintos de V. O conjunto de vértices de um grafo G é denominado $V(G)$, e o conjunto de arestas é denominado $E(G)$.

Se $a = \{u, v\}$ é uma aresta, dizemos que a *incide* a u e v, que u e v são seus *extremos* e que u e v são *adjacentes*. Algumas vezes, para simplificar notação, denotamos uma aresta $\{u, v\}$ por uv. O *grau* $d(v)$ de um vértice v é o número de arestas incidentes a v. Denotamos por $V_i(G)$ o conjunto de vértices de grau i em G, e por $\bar{V}_i(G)$ o conjunto de vértices de grau pelo menos i em G. Um vértice com grau igual a 1 é chamado de *folha*. Um grafo G é *k-regular* se $d(v) = k$ para todo $v \in V$.

Se S é um conjunto de vértices de um grafo $G = (V, E)$, denotamos por $A(S)$ o conjunto das arestas de G com ambos os extremos em S e por $\delta(S)$ o conjunto das arestas de G com um extremo em S e o outro em $V \backslash S$. Um conjunto da forma $\delta(S)$, onde S e $V \backslash S$ não são vazios, é chamado de *corte*.

Um grafo $H = (W, Y)$ é um *subgrafo* de $G = (V, E)$ se $W \subseteq V$ e $Y \subseteq E$. Neste caso, dizemos que G contém H. Se $|W| = |V|$ então dizemos que H é um subgrafo *gerador* de G. Se $F \subseteq E$, então o subgrafo de G com conjunto de arestas F e vértices consistindo dos extremos das arestas de F é o subgrafo *gerado* ou *induzido* por F e denotado por $G[F]$.

Se $M \subseteq V$, então o subgrafo de G com vértices M e conjunto de arestas $A(M)$ é chamado de subgrafo de G induzido por M e denotamos por $G[M]$.

Um *passeio* em G é uma sequência finita não vazia $P = v_0, v_1, \ldots, v_k$, tal que, para todo $1 \leq i \leq k$, v_{i-1} e v_i são vértices adjacentes de G. Se os vértices em P são distintos, então P é chamado de *caminho*. Um *ciclo* $C = v_0, v_1, \ldots, v_k$ é um passeio onde $v_0 = v_k$, $k \geq 2$ e os vértices $v_0, v_1, \ldots, v_{k-1}$ são distintos. Um grafo é *acíclico* se não contém ciclos.

Dois vértices u e v de G estão *conectados* se existe um caminho que começa em u e termina em v. Um grafo é conexo se possui um caminho entre quaisquer dois de seus vértices. Denominamos *componentes* os subgrafos conexos maximais de um grafo. Um grafo acíclico e conexo é chamado de *árvore*. Grafos acíclicos não necessariamente conexos são chamados de *florestas*. Seja G um grafo conexo. Uma subárvore de G é um subgrafo conexo e acíclico de G.

Um grafo é *bipartido* se seu conjunto de vértices pode ser particionado em dois subconjuntos X e Y, tal que, cada aresta tem um extremo em um vértice de X e o outro em um vértice de Y.

Um grafo *orientado* $D = (V, A)$ consiste de um conjunto não vazio V de vértices e um conjuntos A de arcos. Um *arco* é um par ordenado de elementos de V. O conjunto de vértices de um grafo orientado D é denominado $V(D)$, e o conjunto de arcos é denominado $A(D)$. O *grau de saída* de um vértice v é o número de arcos ij de G tal que $i = v$. O *grau de entrada* de v é o número de arcos ij de G tal que $j = v$. Uma *árvore orientada enraizada em s* é um grafo orientado conexo, acíclico, onde o vértice s, que é chamado de raiz, possui grau de entrada igual a zero e existe um caminho de s para todos os outros vértices da árvore.

2.4 Casos Particulares

Nesta seção apresentamos alguns casos particulares do PAGMF encontrados na literatura. Um dos casos particulares foi proposto por Storer [25] e considera o PAGMF apenas em grafos cúbicos, ou seja, grafos em que todos os vértices têm grau igual a 3. Sabe-se que o problema é NP-difícil [15] para grafos cúbicos e recentemente Bonsma mostrou que é APX-difícil [2]. Para essa variação do problema, o trabalho de Lorys e Zwoniak [17] mostra um algoritmo de razão de aproximação 7/4. Em Correa, Fernandes, Matamala e Wakabayashi [6], os autores apresentam um algoritmo 5/3-aproximado. O resultado foi melhorado por Bonsma e Zickfeld [3] para 3/2. Os trabalhos de Lorys e Zwoniak [17] e Bonsma e Zickfeld [3] são apresentados com mais detalhes no Capítulo 4.

Uma variação para grafos bipartidos também tem sido estudada. Considere um dado grafo bipartido $G = (V, E)$ e dois conjuntos de vértices que particionam G, digamos X e Y. O problema de encontrar uma árvore geradora tal que o número de folhas contidas em

X é máxima foi proposto por Rahman e Kaykobad [22]. Li e Toulpise [16] mostraram que o problema é NP-difícil. O trabalho de Focus e Monti [11] considera grafos k-regulares bipartidos. Eles provam que o problema é NP-difícil para qualquer $k \geq 4$ e apresentam um algoritmo guloso cuja razão de aproximação é $2 - 2/(k-1)^2$. Também mostram que para grafos cúbicos bipartidos o algoritmo tem razão de aproximação 2 e incluindo uma etapa de otimização local a razão de aproximação melhora para $1,5$.

Tem-se outro caso particular ao considerar apenas grafos orientados. Nesse caso o que se procura é uma árvore orientada enraizada em s com o número máximo de folhas. As folhas são os vértices que possuem grau de saída igual a zero. Drescher e Vetta [8] apresentam um algoritmo $O(\sqrt{OPT})$-aproximado para esse caso. Daligault e Thomassé [7] apresentam um algoritmo de aproximação com fator de aproximação igual a 92. Schwartges, Spoerchase e Wolff [23] mostraram dois algoritmos aproximados com fator de aproximação de 4 e 2 para grafos orientados acíclicos.

2.5 Problema do conjunto conexo dominante mínimo

Um problema relacionado ao PAGMF é o Problema do Conjunto Conexo Dominante Mínimo (PCCDM). Um conjunto de vértices de um grafo é dito *dominante*, se todo vértice que não esteja no conjunto é adjacente a algum vértice do conjunto. O conjunto é dito *conexo* quando induz um subgrafo conexo. O PCCDM consiste em encontrar um conjunto conexo dominante com o menor número possível de vértices.

A partir de um conjunto conexo dominante mínimo S, pode-se computar de maneira eficiente uma árvore geradora com número máximo de folhas. Basta encontrar uma árvore geradora T de $G[S]$, e adicionar os vértices que não estão em S como folhas de T. Além disso, dado uma árvore geradora com número máximo de folhas T, um conjunto conexo dominante mínimo S é facilmente calculado. O conjunto S é definido pelo vértices em T que não são folhas.

Isto mostra que do ponto de vista de otimização ambos problemas são equivalentes. Note, entretanto, que os resultados sobre a inaproximabilidade deles são bem distintos. Não existe algoritmo de aproximação constante para o PCCDM a menos que $NP \subseteq DTIME(n^{O(log\ log\ n)})$ [14].

Capítulo 3

Algoritmos Aproximados

Neste capítulo apresentamos algoritmos aproximados para o PAGMF. Na primeira seção apresentamos um algoritmo baseado em otimizações locais e provamos sua razão de aproximação de 5. Na seção seguinte apresentamos um algoritmo guloso com razão de aproximação 3 e complexidade de tempo quase linear. Na última seção mostramos um algoritmo 2-aproximado com a melhor razão de aproximação conhecida.

3.1 Algoritmo 5-aproximado

Nesta seção apresentamos algumas definições e um algoritmo 5-aproximado proposto por Lu e Ravi em [18] com complexidade de tempo de $O(n^4)$.

O algoritmo encontra uma árvore geradora inicial e faz mudanças locais enquanto aumenta o número de folhas na árvore geradora resultante até que não seja possível melhorar a solução. As mudanças locais trocam arestas que estão na árvore por arestas que não estão na árvore. Variando o número de arestas que podem participar de cada operação de mudança local é possível ter uma série de algoritmos de aproximação.

3.1.1 Definições

Seja G um grafo conexo e T uma árvore geradora de G. No resto do capítulo supomos que G tem mais de um vértice. Um vértice *interno* de T é um vértice de $\bar{V}_2(T)$. Um vértice de *grau alto* de T é um vértice de $\bar{V}_3(T)$. Uma folha de T é *especial* se ela é adjacente em T a um vértice de grau alto de T. Uma folha de T é *normal* se ela não é especial. Usamos $V_{1E}(T)$ para denotar o conjunto de folhas especiais de T, e $V_{1N}(T)$ para denotar o conjunto de folhas normais de T. Usamos $cam_T(u, w)$ para denotar o caminho que conecta u e v em T. Dizemos que v é *coberto* em T por uma aresta uw em $E(G)\backslash E(T)$ se $v \neq u$, $v \neq w$ e v está em $cam_T(u, w)$.

Um *2-caminho* de T é um caminho maximal em T que contém apenas vértices de $V_2(T)$. Um 2-caminho P de T é *curto* se $|V(P)| = 1$; caso contrário ele é *longo*. Usamos $P_C(T)$ para denotar o conjunto de 2-caminhos curtos de T, e $P_L(T)$ para o conjunto de 2-caminhos longos de T.

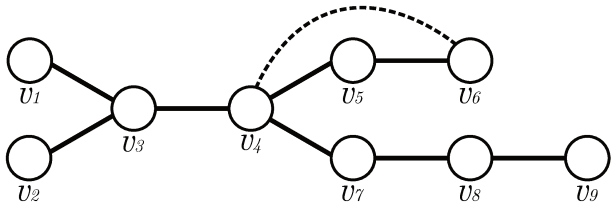


Figura 3.1: Exemplo de folhas normais e especiais, vértice coberto e 2-caminhos.

Na Figura 3.1 apresentamos uma árvore T para exemplificar algumas das definições. As arestas escuras pertence a árvore, e a aresta pontilhada pertence a $E(G)\backslash E(T)$. Os vértices v_1 e v_2 são exemplos de folhas especiais. Já os vértices v_6 e v_9 são folhas normais. O vértice v_5 é coberto pela aresta v_4v_6 e também é um 2-caminho curto. Os vértices v_7 e v_8 pertencem a um 2-caminho longo.

Uma *região* de T é um componente conexo do subgrafo de T obtido pela remoção de todos os vértices em $P_L(T)$. Uma região de T é *especial* se ela contém uma folha especial de T. Uma região é *normal* se não é especial. Sejam R e R' duas regiões distintas de T. O 2-caminho longo de T que conecta R e R' em T, se existir, é chamado de 2-caminho longo *incidente* a R e R'. O número de 2-caminhos longos incidentes a uma região de T é o *grau* da região. Uma região é *interna* se tem grau pelo menos dois. Uma região é *externa* se não é interna. Usamos $R_{iE}(T)$ para denotar o conjunto de regiões especiais de T que tem grau i, e $R_{iN}(T)$ para denotar o conjunto de regiões normais de T que tem grau i. Sejam $R_E(T)$ e $R_N(T)$ os conjuntos de regiões especiais e normais de T respectivamente. Na figura 3.2 a região R_1 é interna especial e as regiões R_2 e R_3 são externas normais.

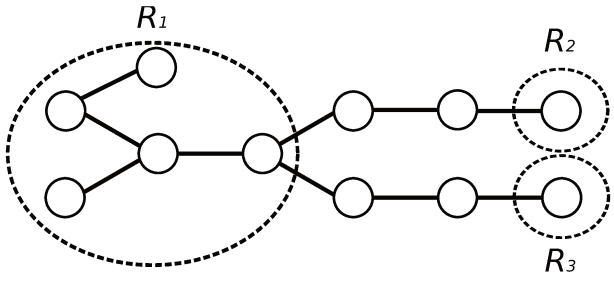


Figura 3.2: Exemplo de regiões.

A seguir destacamos algumas relações referentes aos conjuntos que definimos. Seja G um grafo conexo e T uma árvore geradora de G. Limitando o número de vértices de grau

maior do que ou igual a 3 pelo número de folhas em T, temos a seguinte inequação:

$$|\bar{V}_3(T)| \leq |V_1(T)| - 2. \tag{3.1}$$

Por definição, sabemos que 2-caminhos são adjacentes a folhas normais ou a vértices em $\bar{V}_3(T)$. Então temos a seguinte inequação:

$$|P_L(T)| + |P_C(T)| \leq |\bar{V}_3(T)| + |V_{1N}(T)| - 1. \tag{3.2}$$

O número de regiões de T é exatamente igual a um mais o número de 2-caminhos longos de T. A soma dos graus das regiões é duas vezes a quantidade de 2-caminhos longos de T. Ou seja,

$$|P_L(T)| = |R_E(T)| + |R_N(T)| - 1, \tag{3.3}$$

$$2|P_L(T)| = \sum_{i \geq 0} i(|R_{iE}(T)| + |R_{iN}(T)|). \tag{3.4}$$

Cada região especial de T tem pelo menos uma folha especial. Então

$$|R_E(T)| \leq |V_{1E}(T)|. \tag{3.5}$$

3.1.2 Algoritmo

Seja G um grafo conexo e T uma árvore geradora de G. Seja e uma aresta de T e e' uma aresta de $E(G) \backslash E(T)$. Se $T' = T - e + e'$ ainda é uma árvore geradora de G, então chamamos de *1-mudança* a troca das arestas e e e', e denotamos por (e, e'). Uma 1-mudança (e, e') para T é uma *1-melhoria* se o número de folhas de T' é maior do que o de T. Uma *1-árvore localmente ótima* (1-ALO) de um dado grafo G é uma árvore geradora de G que não admite uma 1-melhoria.

```
   Entrada: G um grafo conexo
   Saída: T uma 1-ALO de G
1  início
2  |  encontre uma árvore geradora T de G;
3  |  enquanto existe uma 1-melhoria (e, e') para T faça
4  |  |  T := T − e + e'.
5  |  fim
6  fim
```

Algoritmo 1: Arvore1LocalmenteOtima (G)

O Algoritmo 1 usa 1-mudança, tem custo de tempo $O(n^4)$ e razão de aproximação 5. Ele começa com uma árvore geradora arbitrária, que pode ser encontrada em tempo

$O(n^2)$, onde n é igual ao número de vértices de G. Então enquanto for possível, aplica 1-melhorias até encontrar uma árvore 1-ALO. O custo para verificar se existe uma 1-melhoria é $O(n^3)$. A troca das arestas de T pode ser feita em tempo constante. O número máximo de iterações do laço é n, uma vez que cada iteração aumenta o número de folhas de T, e T pode ter no máximo $n-1$ folhas. Portanto o custo do algoritmo é de $O(n^4)$ para obter uma 1-ALO.

Resta mostrar que o Algoritmo 1 tem razão de aproximação 5. Antes disso, precisamos de alguns fatos. Apresentamos três propriedades de árvores localmente ótimas. Essas propriedades estão ilustradas na Figura 3.3, onde as arestas escuras são da árvore T.

P1 Seja uw uma aresta em $E(G)\backslash E(T)$, onde u é um vértice interno de T. Então $cam_T(u,w)$ não contém dois vértices consecutivos em $V_2(T)$.

P2 Seja u uma folha normal de T e seja w um vértice interno de T. Então a aresta uw não está em $E(G)\backslash E(T)$.

P3 Seja uw uma aresta em $E(G)\backslash E(T)$, onde u e w são vértices internos de T. Então $cam_T(u,w)$ não contém nenhum vértice de $V_2(T)$.

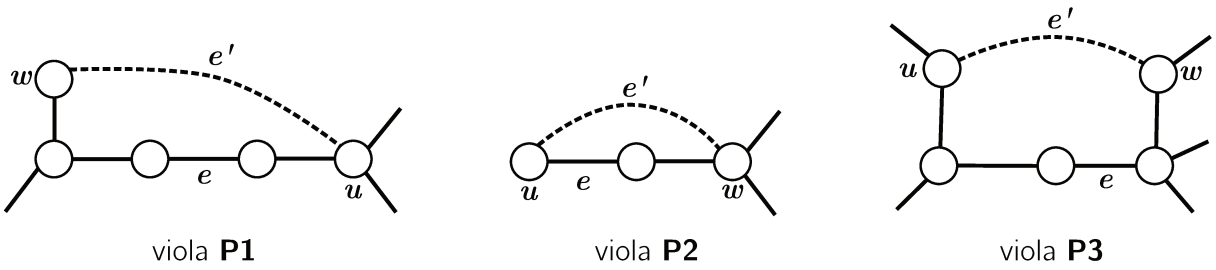


Figura 3.3: Propriedades de 1-ALO.

Essas propriedades são de fácil verificação e omitimos suas provas.

Lema 3.1. *Seja T uma 1-ALO de um grafo G. Seja T' uma árvore geradora de G. Então todo 2-caminho de T contém no máximo quatro folhas de T'.*

Prova. Suponha por contradição que um 2-caminho P de T tem mais de quatro folhas de T'. Sejam v_1, v_2, v, v'_1, e v'_2 cinco folhas de T' em ordem ao longo de P. Seja Q um caminho mais curto em T' de v a um vértice em $V(T')-V(P)$. Seja u um extremo de Q, tal que $u \neq v$. Seja w o vértice mais próximo de u em Q. Veja a Figura 3.4. Note que w está em P e a aresta uw pertence a $E(T')\backslash E(T)$. Por P3 sabemos que $E(T')\backslash E(T)$ não tem aresta incidente a dois vértices distintos de P. Então w está em $cam_T(v_2,v'_1)-\{v_2,v'_1\}$. Isto contraria a 1-otimalidade por P1. □

Mostramos a seguir que $|V_1(T')| \leq 10|V_1(T)|$ para qualquer 1-ALO T de G e qualquer árvore geradora T' de G. Temos que

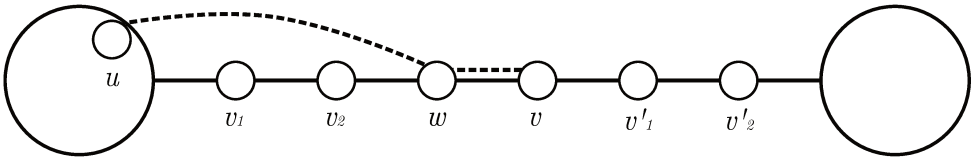


Figura 3.4: Prova do Lema 3.1.

$$\begin{aligned}
|V_1(T')| &= |V_1(T') \cap V_1(T)| + |V_1(T') \cap V_2(T)| + |V_1(T') \cap \bar{V}_3(T)| \\
&\leq |V_1(T)| + |V_1(T') \cap V_2(T)| + |\bar{V}_3(T)| \\
&\leq |V_1(T)| + 4(|P_C(T)| + |P_L(T)|) + |\bar{V}_3(T)| \\
&\leq |V_1(T)| + 4(|V_{1N}(T)| + |\bar{V}_3(T)|) + |\bar{V}_3(T)| \\
&\leq 10|V_1(T)|,
\end{aligned}$$

onde a primeira igualdade vem das definições de $V_i(T)$ e $\bar{V}_i(T)$, a segunda desigualdade segue do Lema 3.1, a terceira desigualdade segue de (3.2) e a última desigualdade de (3.1).

3.1.3 Garantia do fator de aproximação das 1-ALO

Nesta subseção provamos o seguinte teorema.

Teorema 3.2. *Seja T_0 uma 1-ALO de um grafo G_0 e T_0' uma árvore geradora de G_0. Então $|V_1(T_0')| \leq 5|V_1(T_0)|$.*

Para provar o teorema anterior *aumentamos* G_0 e T_0 como segue. Para toda folha normal v de T_0 adjacente a um 2-caminho curto, subdividimos a aresta de T_0 incidente a v e inserimos um pseudo-vértice de grau 2 nesta aresta.

Lema 3.3. *Seja G_0 um grafo e T_0 uma 1-ALO de G_0. Sejam G e T as versões aumentadas de G_0 e T_0, respectivamente. Então T é uma 1-ALO de G.*

Prova. Suponha por contradição que T não é uma 1-ALO de G. Então existe uma 1-melhoria para T que envolve a adição de uma aresta e que pertence a $E(G)\backslash E(T)$. Já que T_0 é uma 1-ALO de G_0, um pseudo-vértice v deve ser coberto por e em T. Para cobrir v, e deve ser incidente a uma folha normal de T_0 adjacente a v. Por P2 sabemos que o outro extremo de e também é uma folha de T_0. Porém, uma aresta incidente a duas folhas não pode participar de uma 1-melhoria. Contradição com a existência de uma 1-melhoria para T. Logo, T é uma 1-ALO. □

Dizemos que T é uma 1-ALO *aumentada* de G se T e G são versões aumentadas de T_0 e G_0, onde T_0 é uma 1-ALO de G_0. Note que por definição uma 1-ALO aumentada não

tem folha normal adjacente a 2-caminho curto. Então toda 1-ALO aumentada T tem a seguinte propriedade:

$$|V_{1N}(T)| = |R_{1N}(T)| \leq |R_N(T)|. \tag{3.6}$$

O seguinte lema garante que se provarmos que o fator de aproximação de uma 1-ALO aumentada é cinco, então o Teorema 3.2 segue do Teorema 3.5.

Lema 3.4. *Seja T_0 uma 1-ALO de G_0. Sejam G e T as versões aumentadas de G_0 e T_0, respectivamente. Então $|V_1(T)| = |V_1(T_0)| \leq |V_1(T_0^*)| \leq |V_1(T^*)|$ onde T_0^* é uma árvore geradora com número máximo de folhas de G_0 e T^* é uma árvore geradora com número máximo de folhas de G.*

Prova. A igualdade é verdadeira porque o aumento não altera o número de folhas em T_0. A desigualdade $|V_1(T_0)| \leq |V_1(T_0^*)|$ segue da definição de T_0^*. A última desigualdade também é verdadeira. Basta observar que temos uma árvore geradora de G direto da versão aumentada de T_0^* com pelo menos $|V_1(T_0^*)|$ folhas. □

Segue do lema que

$$\frac{|V_1(T_0^*)|}{V_1(T_0)} \leq \frac{|V_1(T^*)|}{V_1(T)}.$$

Então para provarmos o Teorema 3.2 é suficiente provar o seguinte teorema.

Teorema 3.5. *Seja T uma 1-ALO aumentada de G, onde G é um grafo. Então $|V_1(T')| \leq 5|V_1(T)|$ para qualquer árvore geradora T' de G.*

Para o resto da seção suponha que T é uma 1-ALO aumentada de G. Seja T' uma árvore geradora de G. Uma região R de T é *inválida* em T' se ou (i) R é uma região especial de T e cada folha especial de T em R é uma folha de T', ou (ii) R é uma região interna normal de T. Uma região de T é *válida* em T' se ela não é inválida em T'.

Lema 3.6. *Seja T uma 1-ALO aumentada de G. Seja T' uma árvore geradora de G. Então o número de folhas de T' que estão em 2-caminhos longos de T é no máximo*

$$4|R_E(T)| - 2r + 3|V_{1N}(T)| - |V_{1N}(T) \cap V_1(T')|,$$

onde r é o número de regiões especiais de T que são inválidas em T'.

Antes de mostrar a prova do Lema 3.6, que é complicada, mostraremos como usá-lo para provar o Teorema 3.5.

Prova do Teorema 3.5. Seja r o número de regiões especiais de T que são inválidas em T'. Então há $|R_E(T)| - r$ regiões especiais de T que são válidas em T'. Por definição, cada região especial de T que é válida em T' tem pelo menos uma folha especial de T que não é folha de T'. Então há pelo menos $|R_E(T)| - r$ folhas especiais de T que não são folhas de T'. Logo

$$|V_1(T') \cap V_{1E}(T)| \leq |V_{1E}(T)| - (|R_E(T)| - r). \tag{3.7}$$

Segue-se que

$$\begin{aligned}
|V_1(T')| &= |V_1(T') \cap V_1(T)| + |V_1(T') \cap V_2(T)| + |V_1(T') \cap \bar{V}_3(T)| \\
&\leq |V_1(T') \cap V_1(T)| + |V_1(T') \cap V_2(T)| + |\bar{V}_3(T)| \\
&= |V_1(T') \cap V_{1E}(T)| + |V_1(T') \cap V_{1N}(T)| + |V_1(T') \cap V(P_C(T))| + \\
&\quad |V_1(T') \cap V(P_L(T))| + |\bar{V}_3(T)| \\
&\leq |V_1(T') \cap V_{1E}(T)| + |V_1(T') \cap V_{1N}(T)| + |P_C(T)| + \\
&\quad |V_1(T') \cap V(P_L(T))| + |\bar{V}_3(T)| \\
&\leq |V_1(T') \cap V_{1E}(T)| + |V_1(T') \cap V_{1N}(T)| + |P_C(T)| + \\
&\quad (4|R_E(T)| - r + 3|V_{1N}(T)| - |V_{1N}(T) \cap V_1(T')|) + |\bar{V}_3(T)| \\
&= |V_1(T') \cap V_{1E}(T)| + |P_C(T)| + \\
&\quad 4|R_E(T)| - r + 3|V_{1N}(T)| + |\bar{V}_3(T)| \\
&\leq |V_{1E}(T)| - (|R_E(T)| - r) + |P_C(T)| + \\
&\quad 4|R_E(T)| - r + 3|V_{1N}(T)| + |\bar{V}_3(T)| \\
&= |V_{1E}(T)| + |P_C(T)| + 3|R_E(T)| + 3|V_{1N}(T)| + |\bar{V}_3(T)| \\
&\leq 3|V_{1E}(T)| + |P_C(T)| + |R_E(T)| + 3|V_{1N}(T)| + |\bar{V}_3(T)| \\
&= |P_C(T)| + |R_E(T)| + 3|V_1(T)| + |\bar{V}_3(T)| \\
&\leq |P_C(T)| + |P_L(T)| - |R_N(T)| + 1 + 3|V_1(T)| + |\bar{V}_3(T)| \\
&\leq |\bar{V}_3(T)| + |V_{1N}(T)| - |R_N(T)| + 3|V_1(T)| + |\bar{V}_3(T)| \\
&\leq 2|\bar{V}_3(T)| + 3|V_1(T)| \\
&\leq 5|V_1(T)|,
\end{aligned}$$

onde as igualdades e desigualdades podem ser obtidas da seguinte maneira. Temos a primeira igualdade das definições de V_i e $\bar{V}_i$. A primeira desigualdade por $|V_1(T') \cap \bar{V}_3(T)| \leq |\bar{V}_3(T)|$. A segunda igualdade por $V_1(T) = V_{1E}(T) \cup V_{1N}(T)$ e $V_2 = V(P_C(T) \cup P_L(T))$. A terceira desigualdade segue do Lema 3.6. A quarta desigualdade segue de (3.7). A quinta desigualdade segue de (3.5). A sexta desigualdade segue de (3.3). A sétima desigualdade segue de (3.2). A oitava desigualdade segue de (3.6). A última desigualdade segue de (3.1). □

Na próxima subseção apresentamos uma prova para o Lema 3.6.

3.1.4 Custo de Regiões e 2-caminhos Longos Deficientes

Seja R uma região de T. O *custo* de R em T', denotado por $custo_{T'}(R)$, é definido como

$$custo_{T'}(R) = \begin{cases} 2 \text{ se } R \text{ é uma região especial de } T \\ 0 \text{ se } R \text{ é uma folha normal de } T \text{ e folha de } T' \\ 1 \text{ caso contrário.} \end{cases}$$

A seguir apresentamos uma versão refinada do Lema 3.1 onde o limite para $|V_1(T') \cap V(P_L(T))|$ é melhor do que $4|P_L(T)|$.

Lema 3.7. *Seja T uma 1-ALO aumentada de G. Seja T' uma árvore geradora de G. Seja P um 2-caminho longo de T incidende as regiões R e R' de T. Então P tem no máximo $custo_{T'}(R) + custo_{T'}(R')$ folhas de T'.*

Seja G, T, T', P, R e R' como definido no Lema 3.7. Um 2-caminho longo P é *deficiente* em T' se $custo_{T'}(R) + custo_{T'}(R') \geq 2$, e P tem no máximo $custo_{T'}(R) + custo_{T'}(R') - 2$ folhas de T'.

O lema seguinte relaciona o número de regiões inválidas e a quantidade de 2-caminhos longos deficientes.

Lema 3.8. *Seja T uma 1-ALO aumentada de G. Seja T' uma árvore geradora de G. Seja k o número de regiões de T que são inválidas em T'. Então existem pelo menos $k-1$ 2-caminhos longos de T distintos que são deficientes em T'.*

Agora apresentamos a prova do Lema 3.6.

Prova do Lema 3.6. Seja $d(R)$ o grau da região R de T. Seja k igual ao número de regiões de T que são inválidas em T' e $l = \sum_{R \in R_E(T) \cup R_N(T)} custo_{T'}(R) d(R)$.

Dos Lemas 3.7 e 3.8 temos que o número de folhas de T' que estão em 2-caminhos longos de T é no máximo $l - 2(k-1)$. Vamos mostrar que $l - 2(k-1)$ é no máximo $4|R_E(T)| - 2r + 3|V_{1N}(T)| - |V_{1N}(T) \cap V_1(T')|$.

Assim,

$$\begin{aligned}
l &= \sum_{R\in R_E(T)} custo_{T'}(R)d(R) + \sum_{R\in R_N(T)-(R_{1N}(T)\cap V_1(T'))} custo_{T'}(R)d(R) \\
&= \sum_{R\in R_E(T)} 2d(R) + \sum_{R\in R_N(T)} d(R) - \sum_{R\in R_{1N}(T)\cap V_1(T')} d(R) \\
&= 2\sum_{i\geq 1} i|R_{iE}(T)| + \sum_{i\geq 1} i|R_{iN}(T)| - |V_{1N}(T)\cap V_1(T')| \\
&= 2\sum_{i\geq 1} i|R_{iE}(T)| + 2\sum_{i\geq 1} i|R_{iN}(T)| - \sum_{i\geq 1} i|R_{iN}(T)| - |V_{1N}(T)\cap V_1(T')| \\
&\leq 4|P_L(T)| - \sum_{i\geq 1} i|R_{iN}(T)| - |V_{1N}(T)\cap V_1(T')| \\
&= 4(|R_E(T)| + |R_N(T)| - 1) - \sum_{i\geq 1} i|R_{iN}(T)| - |V_{1N}(T)\cap V_1(T')|, \qquad (3.8)
\end{aligned}$$

onde a primeira igualdade vem das definições de l e $custo_{T'}(R)$, a segunda é obtida ao substituir os custos das regiões especiais por 2 e separar o segundo somatório, a terceira ao substituirmos $\sum_{R\in R(T)} d(R)$ por $\sum_{i\geq 1} i|R_i(T)|$, a quarta igualdade ao adicionarmos $\sum_{i\geq 1} i|R_{iN}(T)| - \sum_{i\geq 1} i|R_{iN}(T)|$, a desigualdade segue de (3.4) e a última igualdade segue de (3.3).

Por definição uma região R de T é inválida se (i) R é uma região normal interna de T, isto é, $R \in R_N(T) - R_{1N}(T) - R_{0N}(T)$, ou (ii) R é uma região especial de T que é inválida em T'. Logo

$$\begin{aligned}
2(k-1) &= 2(|R_N(T)| - |R_{1N}(T)| - |R_{0N}(T)| + r - 1) \\
&\geq 2(|R_N(T)| - |R_{1N}(T)| + r - 2) \\
&= 2|R_N(T)| - 2|R_{1N}(T)| + 2r - 4, \qquad (3.9)
\end{aligned}$$

onde $R_{0N}(T) \leq 1$. Combinando (3.8), (3.9) e (3.6) temos

$$\begin{aligned}
l - 2(k-1) &\leq 4|R_E(T)| + 2|R_N(T)| - \sum_{i\geq 1} i|R_{iN}(T)| - |V_{1N}(T)\cap V_1(T')| + \\
&\quad 2|R_{1N}(T)| - 2r \\
&\leq 4|R_E(T)| + |R_{1N}(T)| - |V_{1N}(T)\cap V_1(T')| + 2|R_{1N}(T)| - 2r \\
&= 4|R_E(T)| - 2r + 3|V_{1N}(T)| - |V_{1N}(T)\cap V_1(T')|,
\end{aligned}$$

onde a primeira desigualdade é obtida de (3.8) e (3.9), a segunda substituindo $2|R_N(T)| - \sum_{i\geq 1} i|R_{iN}(T)|$ por $|R_{1N}(T)|$ e a última igualdade segue de (3.6) ao substituir $|R_{1N}(T)|$ por $|V_{1N}(T)|$. □

A seguir provamos os Lemas 3.7 e 3.8.

3.1.5 Prova do Lema 3.7

Sejam T uma 1-ALO aumentada de G e T' uma árvore geradora de G. Seja P um 2-caminho longo de T incidente às regiões R e R' de T. Queremos mostrar que P tem no máximo $custo_{T'}(R) + custo_{T'}(R')$ folhas de T'.

Sejam $i = custo_{T'}(R)$ e $j = custoT'(R')$. Suponha por contradição que P tem mais de $i + j$ folhas de T'. Sejam $v_1, \ldots, v_i, v, v'_1, \ldots, v'_j$ as $i + j + 1$ folhas ao longo de P em ordem de R para R'.

Se $i = j = 0$, sabemos que R e R' são folhas de T' e folhas normais de T. Segue que T é um caminho. O vértice v é folha de T' mas é vértice interno de T, logo deve haver alguma aresta em $T - T'$. Por P2 e P3 sabemos que a única possibilidade de aresta em $T - T'$ é a aresta incidente a R e R'. Isso contraria o fato de que T' é uma árvore geradora de G.

Agora analisamos o caso em que $j \geq 1$. Seja Q um caminho em T' de v a um vértice em R'. Seja u o vértice mais próximo a v em Q tal que u não está em P. Seja w o vértice mais próximo de u em $cam_{T'}(u, v)$. A aresta uw está em $T - T'$ e w está em P. Os vértices $v_1, \ldots, v_i, v'_1, \ldots, v'_j$ são folhas em T', então não estão em Q. Por P3 sabemos que não existe aresta em $T - T'$ que é incidente a dois vértices de P. Logo, w está em $cam_T(v_i, v'_1)$.

Vamos mostrar que u não está em R'. Suponha por contradição que u está em R'. O vértice w está em $cam_T(v_i, v'_1)$. Logo uw cobre v'_1. Por P2 e P3 sabemos que u deve ser folha especial de T. Portanto, R' é uma região especial de T, que implica que $j = 2$. Como uw cobre v'_1 e v'_2, a 1-otimalidade de T é contrariada por P1. Então u não está em R'.

Mostramos agora que u não está em R. Suponha por contradição que u está em R. Pelas mesmas razões anteriores sabemos que $i = 0$. Da definição de $custo_{T'}(R)$ sabemos que R, que é u, é folha de T'. Isto contraria o fato de u ter grau maior do que um porque u está em Q, e $u \neq v$ e $u \neq v'$. Logo u não está em R.

Claramente u não está em nenhuma região de T que não seja R ou R', pois uw cobriria todos os vértices de algum 2-caminho longo de T, e assim a 1-otimalidade de T é violada por P1. Além disso, por definição u não está em P.

Segue que u deve estar em algum 2-caminho longo de T diferente de P. Portanto, u e w são vértices internos de T. Isto implica que $i = 0$, pois do contrário uw cobriria v_1 ou v'_1, e assim a 1-otimalidade de T é contrariada por P3. Porém, $i = 0$ significa que R é uma região normal externa que tem apenas uma folha normal de T. Portanto, uw é forçada a cobrir v'_1. Logo a 1-otimalidade de T é contrariada por P3.

Para o caso em que $i \geq 1$, uma contradição pode ser obtida de maneira similar ao caso em que $j \geq 1$. Portanto, P tem no máximo $i + j$ folhas de T' e o lema está provado. □

3.1.6 Prova do Lema 3.8

Seja T uma 1-ALO aumentada de G. Seja T' uma árvore geradora de G. Seja k o número de regiões de T que são inválidas em T'. Queremos mostrar que existem pelo menos $(k-1)$ 2-caminhos longos de T distintos que são deficientes em T'.

Para $k \leq 1$, a prova é trivial. Portanto, suponha que $k \geq 2$. Sejam R_1, ..., R_k as k regiões de T que são inválidas em T'. Vamos identificar $(k-1)$ 2-caminhos longos distintos, denotados por P_1, ..., P_{k-1}, de T que são deficientes em T'.

Seja R uma região de T. Seja $adj(R)$ o subconjunto máximo de $V(P_L(T))$ tal que cada vértice em $adj(R)$ é extremo de uma aresta de T que é incidente a um vértice de R.

Afirmação 1. Seja R uma região de T que é inválida em T'. Seja u um vértice de grau alto em R, e w um vértice que não está em $adj(R) \cup R$. Então $cam_{T'}(u, w)$ tem uma aresta de T' que é incidente a um vértice de $adj(R)$ e um vértice que não está em $adj(R) \cup R$.

Prova. Seja $P' = cam_{T'}(u, w)$. É claro que P' tem uma aresta $u'w'$ tal que u' pertence a $adj(R) \cup R$ e w' não pertence a $adj(R) \cup R$. Queremos mostrar que u' pertence a $adj(R)$. Para isso, primeiro mostramos que u' é um vértice interno de T.

Por definição de regiões inválidas, R ou é uma região normal interna de T ou é uma região especial de T que é inválida em T'. Já que T é aumentada, R não tem folha normal de T. Por definição, $adj(R)$ tem apenas vértices internos de T. Portanto, u' não é uma folha normal de T.

Como R é inválida em T', uma folha especial de T em R, se houver, deve ser uma folha de T'. Sabemos que P' não tem folha especial de T em R, já que u não é uma folha de T e w não pertence a R. Portanto, u' não é uma folha especial de T. Então u' é vértice interno de T.

Suponha por contradição que u' pertence a R. Como existe $cam_T(u', w')$, então a aresta $u'w'$ cobre um vértice de $adj(R)$. Por P3 sabemos que w' é uma folha de T. Assim, w' pertence a uma região de T diferente de R. Isto contraria a 1-otimalidade de T por P1, já que $u'w'$ cobre todos os vértices de algum 2-caminho longo de T. Então u' não pertence a R. Portanto u' pertence a $adj(R)$. □

É fácil verificar que cada região interna R_i tem pelo menos um vértice interno de T. Seja v_i um vértice interno de T que pertence a R_i para todo $1 \leq i \leq k$. Seja $P'_i = cam_{T'}(v_i, v_k)$, para todo $1 \leq i \leq k-1$. Claramente, v_k não pertence a $adj(R_i) \cup R_i$ para todo $1 \leq i \leq k-1$. Pela Afirmação 1, sabemos que P'_i tem uma aresta incidente a um vértice em $adj(R_i)$ e um vértice que não pertence a $adj(R_i) \cup R_i$. Seja u_iw_i uma aresta, onde u_i pertence a $adj(R_i)$ e w_i não pertence a $adj(R_i) \cup R_i$, a aresta em P'_i tal que u_i é o vértice mais próximo de v_i em P'_i. Seja P_i o 2-caminho longo de T que contém u_i.

Queremos mostrar que P_i, para todo $1 \leq i \leq k-1$, são 2-caminhos longos de T distintos e são deficientes em T'.

Afirmação 2. Para todo $1 \leq i \leq k-1$, P_i é deficiente em T'. Além disso, $P_j \neq P_i$ para todo $1 \leq j \neq i \leq k-1$.

Prova. Já que $u_i \neq v_i$, $u_i \neq v_k$, e u_i está em $cam_{T'}(v_i, v_k)$, sabemos que u_i não é folha de T'. Seja R a outra região de T incidente a P_i. Seja w o vértice mais próximo a u_i em $cam_{T'}(u_i, v_k)$ tal que w não pertence a P_i. Seja uw a aresta em $cam_{T'}(u_i, w)$. Note que u pertence a P_i, e que cada vértice em $cam_{T'}(u_i, u)$ não é folha de T'.

Sabemos que w não está em uma região de T que não seja R ou R_i, já que do contrário uw cobriria os vértices de algum 2-caminho longo de T, e assim a 1-otimalidade de T seria contrariada por P1.

Queremos mostrar que w pertence a $adj(R) \cup R$. Por contradição, suponha o contrário. Se $u = u_i$, então $u_i w_i = uw$ e $w = w_i$. Já que w não pertence a P_i por definição de w e w_i não pertence a $adj(R_i) \cup R_i$ por definição de w_i, sabemos que w não pertence a $adj(R_i) \cup R_i \cup P_i$. Por hipótese, w não pertence a R. Logo, w está em algum 2-caminho longo de T diferente de P_i. Mas uw cobre ou um vértice que pertence a $adj(R_i)$ ou um vértice que pertence a P_i. Isto contraria a 1-otimalidade de T por P3.

Se $u \neq u_i$, então u não pertence a $adj(R_i) \cup R_i$. Sabemos que w não pode pertencer a um 2-caminho longo de T diferente de P_i, já que do contrário uw cobriria ou um vértice que pertence a P_i ou um vértice que pertence a $adj(R)$, o que contraria a 1-otimalidade de T por P3. Já que w não está em P_i por definição, então só pode estar em alguma região de T. Porém, w só pode pertencer a R_i ou R, do contrário cobriria algum 2-caminho longo. Por hipótese, w não pertence a $adj(R) \cup R$, então ele pertence a R_i e uw cobre u_i. Por P2 e P3 sabemos que w é uma folha especial de T, e que R_i é uma região especial de T. Já que R_i é inválida em T', w é uma folha de T'. Isto contraria o fato de que w é vértice interno no $cam_{T'}(v_i, v_k)$, já que $w \neq v_i$ e $w \neq v_k$. Então w pertence a $adj(R) \cup R$.

Por definição de regiões inválidas, sabemos que $custo_{T'}(R_i) \geq 1$. Vamos provar a deficiência de P_i mostrando que o $custo_{T'}(R) \geq 1$ e o número de folhas de T' em P_i é no máximo $custo_{T'}(R) - 1$ para os dois casos seguintes.

O primeiro caso é quando w pertence a $adj(R)$. Já que w não pertence a P_i, w está em algum 2-caminho longo diferente de P_i. Por P3 sabemos que u pertence a $adj(R)$, já que do contrário o vértice em $P_i \cap adj(R)$ seria coberto por uw. Já que u pertence a $adj(R)$ e u pertence a P_i, P_i não tem folha de T'. Agora basta mostrar que $custo_{T'}(R) \geq 1$. Já que w pertence a $adj(R)$ e w não pertence a P_i, então o grau de R é pelo menos 2. Portanto, $custo_{T'}(R) \geq 1$, já que R não é uma região normal externa.

O segunto caso é quando w pertence a R. O vértice w está no interior do $cam_{T'}(v_i, v_k)$, então w não é folha de T'. Portanto, sabemos que $custo_{T'}(R) \geq 1$.

Se u pertence a $adj(R)$, então P_i não tem folha de T', e assim P_i é deficiente em T'. Se u não pertence a $adj(R)$, então uw cobre algum vértice de $adj(R)$. Segue por P2 e P3 que w é uma folha especial de T. Portanto, R é uma região especial de T, e $custo_{T'}(R) = 2$. Agora basta mostrar que P_i tem no máximo uma folha de T'. Suponha por contradição que P_i tem duas folha de T'. Já que todos os vértices que pertencem ao $cam_T(u_i, u)$ não são folhas de T', então as folhas de T' que pertencem a P_i são cobertas por uw. Veja a Figura 3.5. A 1-otimalidade de T é contrariada por P1, já que u é um vértice interno. Então os caminhos $P_i, 1 \leq i \leq k-1$, são deficientes em T'.

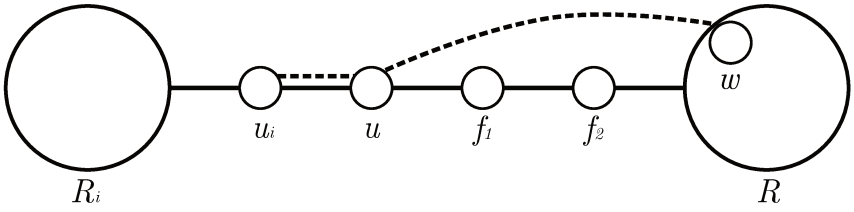


Figura 3.5: Ilustração da demonstração da afirmação 2 no caso que $w \in R$ e $u \notin adj(R)$.

Agora vamos mostrar que $P_i, 1 \leq i \leq k-1$, são distintos. Suponha por contradição que $P_i = P_j$ para algum $1 \leq i \neq j \leq k-1$. Sabemos que P_i é incidente a R_i e P_j é incidente a R_j. Já que $R_i \neq R_j$, sabemos que P_i é incidente a R_i e R_j. Portanto, R é inválida em T', já que $R = R_j$. Como mostramos antes w pertence a $adj(R) \cup R$. Portanto $w \neq u_i$ e $w \neq v_k$. O vértice w não é folha de T' já que pertence ao $cam_{T'}(u_i, v_k)$. Sabemos por definição que R não tem folha normal de T e toda folha especial de R, se existir, é folha em T'. Logo, w é vértice interno de T. Assim, u_j, o único vértice em $P_i \cap adj(R)$, deve ser igual a u, já que do contrário uw cobriria u_j, e a 1-otimalidade de T seria contrariada por P3. Segue que u_j está em P_i, e u_j está no $cam_{T'}(u_i, v_k)$. Trocando R_i e R_j, pelas razões anteriores sabemos que u_i está em P_j e u_i está no $cam_{T'}(u_j, v_k)$. Mas isto contraria o fato de T' ser uma árvore geradora de G. Logo $P_i, 1 \leq i \leq k-1$, são distintos. □

3.2 Algoritmo 3-aproximado

Nesta seção apresentamos um algoritmo guloso proposto por Lu e Ravi em [19] com razão de aproximação 3 e complexidade de tempo quase linear.

O algoritmo encontra uma floresta F de G e adiciona arestas a F para construir uma árvore geradora T de G. A chave para o algoritmo é a maneira que a floresta F é construída. A seguir apresentamos algumas definições que usaremos nesta seção. Em seguida descrevemos o algoritmo e fazemos uma análise da complexidade de tempo. Finalmente, apresentamos uma prova da razão de aproximação.

3.2.1 Definições

Seja G um grafo conexo. Uma subárvore de G é não trivial se tem mais de um vértice. Seja T uma subárvore não trivial de G. Podemos observar que

$$|\bar{V}_3(T)| \leq |V_1(T)| - 2. \tag{3.10}$$

Seja T uma subárvore de G. Dizemos que T é *frondosa* se $\bar{V}_3(T)$ é não vazio e todo vértice em $V_2(T)$ é adjacente a exatamente dois vértices de $\bar{V}_3(T)$. Veja um exemplo de subárvore frondosa na Figura 3.6. Uma floresta F de G é frondosa se F é composta de subárvores frondosas disjuntas de G. Dizemos que F é frondosa maximal se F não é subgrafo de outra floresta frondosa de G.

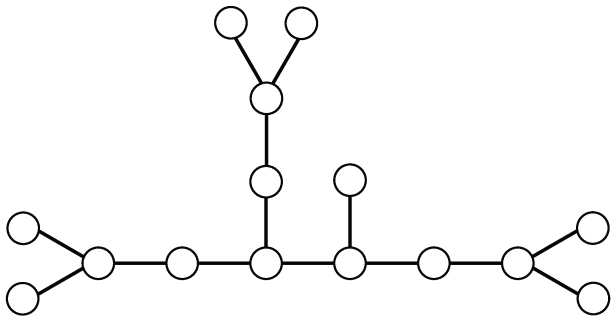


Figura 3.6: Exemplo de subárvore frondosa.

3.2.2 Algoritmo

Dado um grafo conexo G, o algoritmo encontra uma árvore geradora T de G em dois passos. No primeiro, obtém uma floresta frondosa maximal F de G e no segundo, adiciona arestas a F para construir uma árvore geradora T de G.

A seguir apresentamos uma maneira de calcular uma floresta frondosa maximal. Na execução do algoritmo $S(w)$ representa a subárvore de F que atualmente contém o vértice w. O grau do vértice w em F é igual a $d(w)$. A variável d' é o número máximo de arestas adjacentes a v que podem ser adicionadas a F sem criar ciclos. Se $d' + d(v)$ é maior ou igual a 3, então adicionamos as d' arestas a F e unimos as subárvores incidentes a estas

arestas.

```
   Entrada: G um grafo conexo
   Saída: F como uma floresta frondosa maximal de G
 1 início
 2     Seja F um conjunto vazio.
 3     para todo vértice v em G faça
 4         S(v) := {v}.
 5         d(v) := 0.
 6     fim
 7     para todo vértice v em G faça
 8         S' := 0.
 9         d' := 0.
10         para todo vértice u adjacente a v em G faça
11             se u ∉ S(v) e S(u) ∉ S' então
12                 d' := d' + 1.
13                 Inserir S(u) em S'.
14             fim
15         fim
16         se d(v) + d' ≥ 3 então
17             para todo S(u) em S' faça
18                 Adicionar a aresta uv a F.
19                 Unir S(v) e S(u).
20                 Atualizar d(u) := d(u) + 1 e d(v) := d(v) + 1.
21             fim
22         fim
23     fim
24 fim
```

Algoritmo 2: FlorestaFrondosaMaximal (G)

Usando a estrutura de dados de *union-find* para as operações de conjuntos, para referência veja o livro do Cormen, Leiserson, Rivest e Stein [5], podemos encontrar uma floresta frondosa maximal usando o Algoritmo 2 em tempo $O((m+n)\alpha(m,n))$, onde n é igual ao número de vértices e m ao número de arestas.

O segundo passo que é adicionar arestas a F para construir uma árvore geradora de G é trivial e pode ser feito em tempo $O(m+n)$. Logo a complexidade de tempo do algoritmo é $O((m+n)\alpha(m,n))$.

3.2.3 Garantia da razão de aproximação

Nesta subseção provamos o seguinte teorema.

Teorema 3.9. *Seja F uma floresta frondosa maximal de G. Seja T uma árvore geradora de G tal que F é subgrafo de T. Então*

$$|V_1(T')| \leq 3|V_1(T)|$$

para qualquer árvore geradora T' de G.

Os três lemas seguintes são usados para provar o teorema anterior.

Lema 3.10. *Seja T uma subárvore frondosa de G. Então $|V(T)| \leq 3|V_1(T)| - 5$.*

Prova. Já que T é frondosa, todo vértice em $V_2(T)$ é adjacente a exatamente 2 vértices em $\bar{V}_3(T)$. Considere a subárvore onde cada vértice em $V_2(T)$ é substituído por uma aresta e as folhas são retiradas. Veja na Figura 3.7 um exemplo de uma árvore frondosa T antes e depois da retirada das folhas e substituição dos vértices em $V_2(T)$ por arestas. Portanto, o número de vértices de grau 2 em T é no máximo o número de vértices da subárvore resultante menos um, ou seja,

$$|V_2(T)| \leq |\bar{V}_3(T)| - 1. \tag{3.11}$$

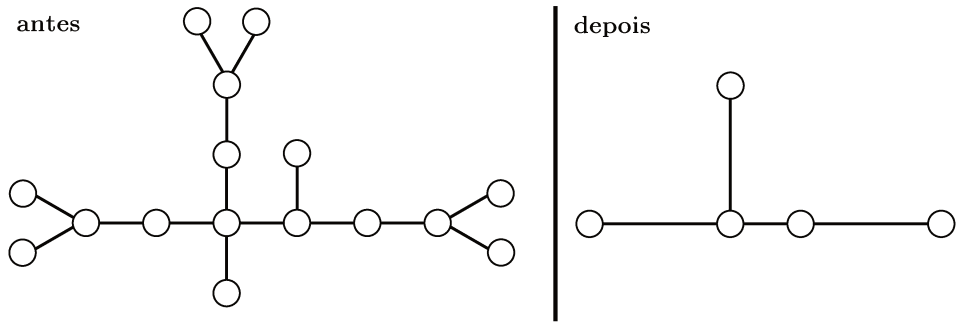


Figura 3.7: Exemplo de árvore frondosa antes e depois da retirada das folhas e substituição dos vértices em $V_2(T)$ por arestas.

Assim,

$$\begin{aligned} |V(T)| &= |V_1(T)| + |V_2(T)| + |\bar{V}_3(T)| \\ &\leq |V_1(T)| + 2|\bar{V}_3(T)| - 1 \\ &\leq |V_1(T)| + 2(|V_1(T) - 2|) - 1 \\ &= 3|V_1(T)| - 5, \end{aligned}$$

onde a primeira igualdade vem das definições de $V_i(T)$ e $\bar{V}_i(T)$, a primeira desigualdade segue de (3.11) e a segunda desigualdade segue de (3.10). □

Seja F uma floresta frondosa maximal de G. Sejam $T_1, \ldots, T_k$ as subárvores frondosas disjuntas de F. A seguir, apresentamos três propriedades de F. Na Figura 3.8 (adaptada de [19]) podemos observar cada uma das propriedades. As linhas escuras são arestas de F e as linhas pontilhadas são arestas de $G - F$.

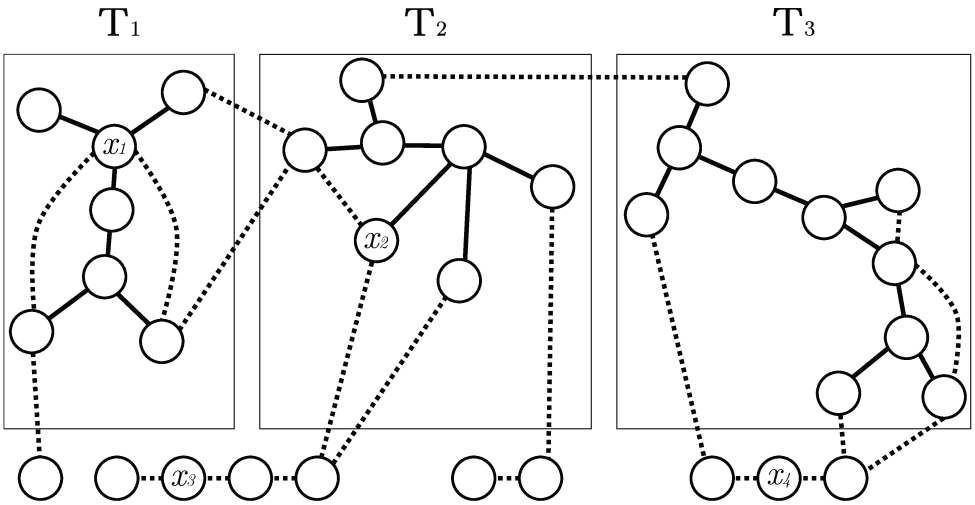


Figura 3.8: Um exemplo de floresta frondosa maximal representado pelas arestas escuras. As linhas pontilhadas são as arestas restantes de G.

Propriedade 1. Seja w um vértice de $\bar{V}_2(T_i)$. Então os vértices adjacentes a w em G pertencem a T_i.

Demonstração. Suponha por contradição que w é adjacente a v em G, tal que v não pertence a T_i. A aresta wv pode ser adicionada a F, então a maximalidade de F é contrariada. □

Propriedade 2. Seja w um vértice de T_i. Sejam w_1 e w_2 dois vértices distintos adjacentes a w em G. Se w_1 não pertence a F, então w_2 pertence a T_i.

Demonstração. Suponha por contradição que w_1 e w_2 não pertencem a F. Já que w pertence a T_i sabemos que o grau de w em T_i é pelo menos 1. Assim, w_1 e w_2 podem ser adicionados a F, contradição com a maximalidade de F. Então w_2 pertence a F. Agora suponha por contradição que w_2 pertence a $T_j, j \neq i$. Já sabemos que o grau de w em T_i é pelo menos 1, então w_1 e as arestas ww_1 e ww_2 podem ser adicionadas a F, contradição com a maximalidade de F. Portanto w_2 pertence a T_i. □

Propriedade 3. Seja w um vértice que não pertence a F. Se w é adjacente a dois vértices que não estão em F, então o grau de w em G é 2.

Demonstração. Suponha por contradição que w tem grau ≥ 3. Então w deveria pertencer a F, contradição com a maximalidade de F. □

O vértice x_1 é um exemplo de w da propriedade 1. O vértice x_2 é um exemplo de w da propriedade 2. Os vértices x_3 e x_4 são exemplos de w da propriedade 3.

Lema 3.11. *Seja F uma floresta frondosa maximal de G que é composta por k árvores frondosas disjuntas $T_1, \ldots, T_k$ de G. Então*

$$|V_1(T')| \leq |V(F)| - k + 1$$

para qualquer árvore geradora T' de G.

Prova. Primeiro vamos mostrar que $|V(F) - V_1(T')| \geq l + k - 1$, onde $l = |V_1(T') - V(F)|$. Depois disso a prova do lema é trivial.

Seja v_i um vértice que pertence a $\bar{V}_3(T_i)$ para todo $i = 1, \ldots, k$. Para todo $i = 1, \ldots, k-1$, seja u_i o vértice em T_i que é o mais distante de v_i em $cam_{T'}(v_i, v_k)$. Seja $v'_1, \ldots, v'_l$ os vértices distintos em $V_1(T') - V(F)$. Para todo $j = 1, \ldots, l$, seja u'_j o vértice em F que é mais próximo de v'_j em $cam_{T'}(v'_j, v_k)$. Queremos mostrar que os vértices $u_1, \ldots, u_{k-1}, u'_1, \ldots, u'_l$ são distintos e que pertencem a $V(F) - V_1(T')$.

Pela definição de u_i, sabemos que em $cam_{T'}(u_i, v_k)$ apenas o vértice u_i pertence a T_i. Segue pela propriedade 1 que u_i pertence a $V_1(T_i)$. Portanto, $u_i \neq v_i$ e $u_i \neq v_k$. Assim, u_i não pertence a $V_1(T')$, já que u_i tem grau maior do que 1 em T'. Então u_i pertence a $V(F) - V_1(T')$. Além disso, como $T_1, \ldots, T_k$ são disjuntas e u_i pertence a T_i para todo $i = 1, \ldots, k-1$, sabemos que $u_1, \ldots, u_{k-1}$ são $(k-1)$ vértices distintos em $V(F) - V_1(T')$.

Por definição v'_j não pertence a $V(F)$ e u'_j pertence a $V(T_{j*})$ para algum T_{j*} que contém u'_j, então $u'_j \neq v'_j$. Sabemos que u'_j e v'_j são vizinhos em G. Pela propriedade 1 u'_j pertence a $V_1(T_{j*})$, então $u'_j \neq v_k$. Logo u'_j não é folha em T', já que é vértice interno em $cam_{T'}(v'_j, v_k)$. Então u'_j pertence a $V(F) - V_1(T')$. Agora vamos mostrar que $u'_1, \ldots, u'_l$ são distintos.

Suponha por contradição que $u'_j = u'_{j'}$ para algum $j \neq j'$. Seja $P = cam_{T'}(u'_j, v'_j)$ e $P' = cam_{T'}(u'_{j'}, v'_{j'})$. Seja w o vértice em $V(P) \cap V(P')$ que é mais próximo de v'_j. Já que v'_j não pertence a P', existe um vértice w_1 no $cam_{T'}(w, v'_j)$ tal que ww_1 é uma aresta de P. Já que $v'_{j'}$ não pertence a P, existe um vértice w_2 em $cam_{T'}(w, v'_{j'})$ tal que ww_2 é uma aresta de P'. Claramente, w_1 e w_2 não pertencem a F e $w_1 \neq w_2$. Veja na Figura 3.9 uma ilustração onde as linhas escuras são arestas e as linhas tracejadas representam caminhos entre seus extremos. Segue pela propriedade 2 que w não pertence a F, o que implica que $w \neq u'_j$. Portanto, w pertence a $\bar{V}_3(G)$ e o fato de F ser frondosa maximal é contrariado pela propriedade 3. Então $u'_1, \ldots, u'_l$ são distintos. Agora vamos mostrar que $u_i \neq u'_j$ para $1 \leq i \leq k-1$ e $1 \leq j \leq l$.

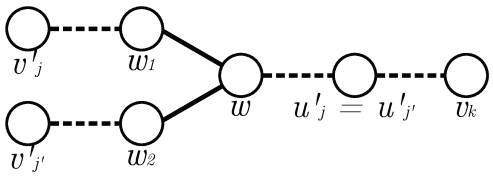


Figura 3.9: Ilustração de w, w_1 e w_2.

Suponha por contradição que $u_i = u'_j$ para algum $i \in \{1, \ldots, k-1\}$ e $j \in \{1, \ldots, l\}$. Seja $P = cam_{T'}(u_i, v_k)$ e $P' = cam_{T'}(u'_j, v'_j)$. Seja w o vértice em $V(P) \cap V(P')$ mais próximo a v'_j. Já que v'_j não pertence a P, existe um vértice w_1 no $cam_{T'}(w, v'_j)$ tal que ww_1 é uma aresta de P. Já que v_k não pertence a P', existe um vértice w_2 no $camT'(w, v_k)$ tal que ww_2 é uma aresta de P'. Claramente, $w_1 \neq w_2$ e por definição de u'_j sabemos que w_1 não pertence a F. Por definição de u'_j e u_j sabemos que $w = u'_j = u_j$ e que w pertence a T_i. Mas pela propriedade 2 temos que w não pertence a T_i, contradição. Então $u_1, \ldots, u_{k-1}, u'_1, \ldots, u'_l$ são $l + k - 1$ vértices distintos que pertencem a $V(F) - V_1(T')$.

Logo,

$$\begin{aligned}
|V(F) - V_1(T')| &\geq l + k - 1 \\
|V(F) - V_1(T')| &\geq |V_1(T') - V(F)| + k - 1 \\
|V(F)| - |V(F) \cap V_1(T')| &\geq |V_1(T')| - |V_1(T') \cap V(F)| + k - 1 \\
|V(F)| &\geq |V_1(T')| + k - 1 \\
|V_1(T')| &\leq |V(F)| - k + 1,
\end{aligned}$$

onde a segunda desigualdade vem da definição de l ao substituir l por $|V_1(T') - V(F)|$ e a terceira desigualdade é obtida a partir do fato de que $|A - B| = |A| - |A \cap B|$, para quaisquer conjuntos A e B. □

Lema 3.12. *Seja F uma floresta de G que tem k árvores não triviais disjuntas. Seja T uma árvore geradora de G tal que F é um subgrafo de T. Então $|V_1(T)| \geq |V_1(F)| - 2(k-1)$.*

Prova. A intuição da prova é que as árvores de F podem ser conectadas até formarem uma árvore geradora da seguinte maneira. Iterativamente adicionando uma aresta incidente a duas árvores desconexas. Cada adição de aresta aumenta o grau de no máximo duas folhas dessas árvores. Seja F' a floresta de G obtida de F pela adição de uma aresta e que pertence a $T - F$. Seja k' o número de subárvores não triviais disjuntas de F'. Vamos mostrar que

$$|V_1(F')| - 2(k' - 1) \geq |V_1(F)| - 2(k - 1). \tag{3.12}$$

O lema segue por indução, já que o número de subárvores não triviais disjuntas em T é 1. Para provar (3.12) temos que analisar os três casos a seguir que representam as combinações de extremos da aresta e.

Primeiro se e não incide a nenhuma subárvore não trivial de F, então a adição de e forma uma nova subárvore com dois vértices. Assim $k' = k + 1$ e $|V_1(F')| = |V_1(F)| + 2$. O segundo caso é se e incide a exatamente uma subárvore não trivial de F, então $k' = k$ e $|V_1(F')| \geq |V_1(F)|$. Por fim, se e incide a duas subárvores de F, então $k' = k - 1$ e $|V_1(F')| \geq |V_1(F)| - 2$. A equação (3.12) se mantém válida para todos os casos, então $|V_1(T)| \geq |V_1(F)| - 2(k-1)$. □

Agora podemos provar o Teorema 3.9.

Prova do Teorema 3.9. Suponha que F tem k subárvores frondosas disjuntas T_i, $1 \leq i \leq k$. Pelo lema 3.10 sabemos que

$$\begin{aligned} |V(F)| &= \sum_{i=1}^{k} |V(T_i)| \\ &\leq 3|V_1(F)| - 5k. \end{aligned}$$

Assim,

$$\begin{aligned} |V_1(T')| &\leq |V(F)| - k + 1 \\ &\leq 3|V_1(F)| - 5k - k + 1 \\ &\leq 3(|V_1(T)| + 2(k-1)) - 6k + 1 \\ &= 3|V_1(T)| - 5 \\ &\leq 3|V_1(T)|, \end{aligned}$$

onde a primeira inequação vem direto do lema 3.11, a segunda pelo lema 3.10 ao substituir $|V(F)|$ por $3|V_1(F)| - 5k$ e a terceira pelo lema 3.12 ao substituir $V_1(F)$ por $|V_1(T)| + 2(k-1)$. □

3.3 Algoritmo 2-aproximado

Nesta seção apresentamos um algoritmo proposto por Solis-Oba em [24] com razão de aproximação 2.

O algoritmo é baseado no algoritmo de Lu e Ravi apresentado na seção 3.2. Primeiro encontra uma floresta frondosa F de G usando regras de expansão e atribuindo prioridade as regras. Em seguida adiciona arestas a F para construir uma árvore geradora T de G. A seguir apresentamos o algoritmo e uma análise da complexidade de tempo. Em seguida algumas definições que usaremos nesta seção e finalmente uma prova que o algoritmo é 3-aproximado. Solis-Oba apresenta uma prova de que o algoritmo é 2-aproximado em [24]. Aparentemente, não há um consenso entre os pesquisadores se a prova da garantia de aproximação está correta. Assim, optamos por não apresentar sua prova aqui. Preferimos mostrar uma prova de que o algoritmo é 3-aproximado.

3.3.1 Algoritmo

Seja G um grafo conexo. O algoritmo contrói uma floresta F com $(k+1)$ árvores $T_i, i = 0, \ldots, k$, e depois conecta as árvore de F e os vértices que não pertencem a F para formar uma árvore geradora T de G.

Descrevemos a seguir como a floresta F é construída. A construção de toda árvore T_i começa pela escolha de um vértice de grau pelo menos 3 que será a raiz da árvore. A raiz é adicionada a T_i como vértice interno e seus vizinhos são adicionado como folhas. Enquanto for possível as regras de expansão são aplicadas às folhas para aumentar a árvore. Definimos as duas regras usadas a seguir:

Regra 1. Se uma folha x tem pelo menos 2 vizinhos que não pertencem a F, então esses vizinhos são incluídos em T_i como filhos de x. Veja a Figura 3.10.

Regra 2. Se uma folha x tem apenas um vizinho y que não pertence a F e y tem pelo menos 2 vizinhos que não pertencem a F, então y é adicionado a T_i como filho de x e os vizinhos de y que não pertencem a F são adicionados a T_i como seus filhos. Veja a Figura 3.11.

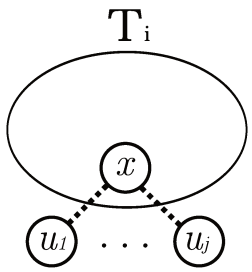


Figura 3.10: Ilustração da regra 1, onde $j \geq 2$.

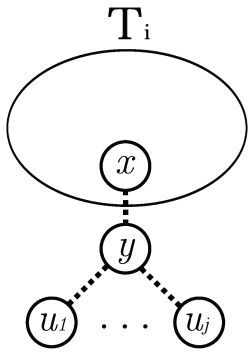


Figura 3.11: Ilustração da regra 2, onde $j \geq 2$.

São definidas prioridades para as expansões como a seguir. A regra 2 quando y tem exatamente 2 vizinhos que não estão em F tem prioridade 1. Todas as outras expansões

tem prioridade 2. Na construção de T_i, se duas folhas de T_i podem ser expandidas, então a folha com a expansão de maior prioridade é expandida primeiro. Apresentamos o algoritmo a seguir.

```
   Entrada: G um grafo conexo
   Saída: T uma árvore geradora de G
 1 início
 2     F := ∅;
 3     i := 0;
 4     enquanto existe um vértice v com grau pelo menos 3 em G faça
 5         Construa uma árvore T_i com raiz v e os vizinhos de v como folhas;
 6         enquanto pelo menos uma folha de T_i pode ser expandida faça
 7             Seja x uma folha de T_i que pode ser expandida com uma regra de
               prioridade mais alta;
 8             Expanda x;
 9         fim
10         F := F ∪ T_i;
11         Remova de G todos os vértices em T_i;
12         i := i + 1;
13     fim
14     Conecte as árvores em F e todos os vértices que não estão em F para formar
       uma árvore geradora T.
15 fim
```

Algoritmo 3: Arvore (G)

É fácil ver que cada passo do algoritmo pode ser executado em tempo polinomial. Além disso, os laços interno e externo executam no máximo $O(n)$ vezes, onde n é o número de vértices de G, pois cada vértice pode ser expandido no máximo uma vez e ao término da construção de cada árvore T_i os vértices de T_i são removidos de G. Logo o algoritmo tem complexidade de tempo polinomial. Para detalhes de implementação consulte a seção 5.2.2.

3.3.2 Definições

Sejam F e T produzidas pelo Algoritmo 3, onde $F = \{T_0, T_1, \ldots, T_k\}$ é a floresta e T é a árvore geradora de um grafo G.

O vértice interno y adicionado pela regra de expansão de prioridade 1 é chamado de vértice preto. Para uma árvore T_i que pertence a F, denotamos por $B(T_i)$ o conjunto de vértices pretos em T_i e por $V(T_i)$ os vértices que pertencem a T_i. O conjunto de vértices pretos em F é denotado por $B(F)$ e o conjunto de vértices de F é denotado por $V(F)$.

Chamamos de vértices exteriores os vértices de G que não pertencem a F. Denotamos por X o conjunto de vértices exteriores. Um vértice exterior não pode ser adjacente a um vértice interno de alguma árvore T_i porque quando um vértice x é expandido todos os seus vizinhos que não pertencem a F são adicionados a T_i.

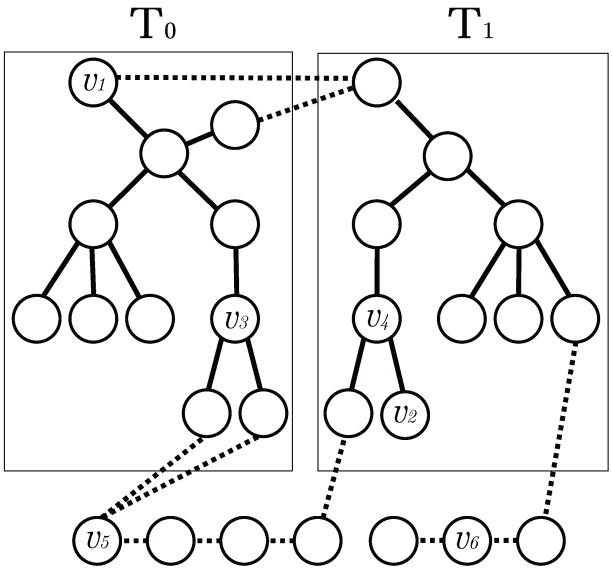


Figura 3.12: Exemplo de uma floresta F construída pelo Algoritmo 3.

Na Figura 3.12 temos um exemplo de uma floresta com duas árvores produzida pelo Algoritmo 3. A arestas escuras são arestas de F e as pontilhadas de $E(G)\backslash E(F)$. A árvore T_0 tem 7 folhas e T_1 tem 6 folhas. Os vértices v_1 e v_2 são exemplos de folhas de F. Os vértices v_3 e v_4 são vértices pretos e v_5 e v_6 são exemplos de vértices exteriores.

3.3.3 Garantia da razão de aproximação

Nesta subseção provamos que o Algoritmo 3 é 3-aproximado apresentando um limite superior para o número de folhas de qualquer árvore geradora e um limite inferior para o número de folhas de uma árvore T produzida pelo Algoritmo 3.

Mostraremos que a floresta F tem a propriedade que exatamente $2k$ folhas são mortas para conectar as $(k+1)$ árvores e formar a árvore geradora T. Para conectar as árvores de F são mortas $2k$ folhas e o número de folhas não é alterado ao conectar cada componente conexo de $G[X]$ que não foi usado para conectar árvores de F. Esses fatos nos permitem mostrar um limitante superior para o número de folhas em qualquer árvore geradora de G em termos do número de vértices em F.

Lema 3.13. *Seja $G' = (V', E')$ o grafo formado pela contração de toda árvore T_i em F*

a um único vértice e removendo as arestas múltiplas entre pares de vértices. Todo vértice exterior tem no máximo grau 2 em G'.

Prova. Suponha por contradição que existe um vértice exterior v que tem grau pelo menos 3 em G'. Considere 3 vizinhos de v. Note que esses 3 vizinhos de v não podem ser vértices exteriores porque o Algoritmo 3 teria escolhido v como raiz de uma nova árvore. Logo, pelo menos um vizinho de v pertence a F. Seja T_i a primeira árvore construída pelo algoritmo que contém um vizinho u de v. Note que pela construção de G' os outros 2 vizinhos considerados de v não pertencem a T_i. Já que v é adjacente a pelo menos dois vértices que não estão em F então o Algoritmo 3 expandiria u pela regra 2 e v seria adicionado a T_i, contradição com o fato de v ser vértice exterior. □

Lema 3.14. *Seja u uma folha de alguma árvore T_i de F. Se u é adjacente a dois vértices v, w que não pertencem a T_i, então v e w são folhas da mesma árvore T_j de F.*

Prova. Primeiro vamos mostrar que v e w não são vértices internos em F. Suponha por contradição que v é vértice interno em alguma árvore T_j. Se o algoritmo constrói a árvore T_j antes da árvore T_i, então u deveria ter sido colocado como filho de v em T_j. Se T_i é contruída primeiro, então a folha u deveria ter sido expandida e teria v como seu filho. De qualquer forma, u e v estariam na mesma árvore, uma contradição. Então, v não é vértice interno de F. O mesmo vale para w.

A seguir faremos uma análise dos possíveis casos restantes. Os vértices v e w não podem ser ambos vértices exteriores, porque se fossem então o vértice u teria sido expandido, mas u é folha em T_i. Pelo menos um de v, w deve ser folha em F. Seja v uma folha de alguma árvore T_j. Mostraremos que w não pode ser vértice exterior e nem pertencer a uma árvore T_k, tal que $k \neq j$.

Suponha que w é um vértice exterior. Se o algoritmo constrói a árvore T_j antes da árvore T_i, então v teria sido expandido e u seria seu filho. Se T_i é contruída primeiro, então u teria sido expandido e v seria seu filho. Em ambos os casos u e v estariam na mesma árvore, uma contradição. Então, w não é um vértice exterior. Logo, v e w são folhas em F.

Suponha que w pertence a uma árvore T_k, tal que $k \neq j$. Veja a Figura 3.13. Por definição de w sabemos que $k \neq i$. As árvores T_i, T_j e T_k foram contruídas pelo Algoritmo 3 em alguma ordem. Se T_i foi a primeira a ser construída, então u seria expandido e v e w seria seus filhos em T_i. Se T_j foi construída primeiro, então v seria expandido e u seria seu filho em T_j. Se T_k foi a primeira, então w seria expandido e u seria seu filho em T_k. Sabemos que u não está na mesma árvore de v ou w. Então v e w pertencem a mesma árvore T_j. □

Corolário 3.15. *Qualquer árvore geradora de G tem no máximo $|V(F)| - 2k$ folhas.*

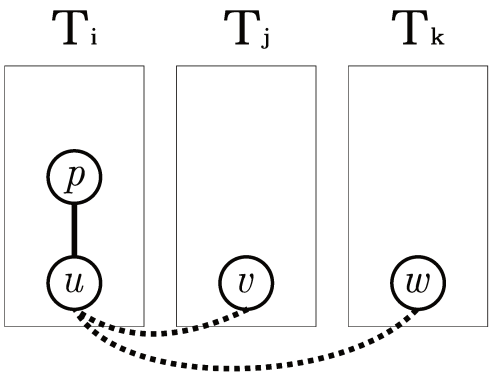


Figura 3.13: Ilustração da prova do Lema 3.14.

Prova. Sejam T^* uma árvore geradora de G e F^* a floresta induzida por T^* em $V(F)$. Seja z o número de vértices exteriores que são folhas em T^*, ou seja, $z = |V(X) \cap V_1(T^*)|$. Um limitante para o número máximo de folhas de T^* é $|V(F)| + z$. Pelos lemas 3.13 e 3.14, qualquer maneira de conectar as árvores de F^* deve matar pelo menos $2k$ folhas de F^*. Além disso, pelo lema 3.13 para conectar cada vértice exterior que é folha em T^* exatamente uma folha de F^* é morta. Logo $V_1(T^*) \leq |V(F)|+z-2k-z = |V(F)|-2k$. □

Agora mostramos um limitante inferior para o número de folhas em uma floresta F e provamos que o Algoritmo 3 é 3-aproximado.

Lema 3.16. *Para qualquer árvore T_i de F, $|V_1(T_i)| \geq 3 + |B(T_i)| + (|V(T_i)| - 3|B(T_i)| - 4)/2$.*

Prova. A raiz de T_i tem grau pelo menos 3. Então quando adicionada a T_i terá pelo menos 3 filhos. A aplicação da expansão de prioridade 1 mata uma folha e adiciona 2. Logo, aumenta 1 folha por aplicação. É aplicada $|B(T_i)|$ vezes. Os outros $(|V(T_i)| - 3|B(T_i)| - 4)$ vértices de T_i são adicionados por expansões de prioridade 2. Note que toda aplicação da regra 1 tem prioridade 2. Uma aplicação da regra 1 adiciona pelo menos 2 vértices a T_i e mata uma folha. Uma aplicação da regra 2 de prioridade 2 adiciona pelo menos 4 vértices a T_i, sendo que um deles não será folha, e mata uma folha. Logo, as expansões de prioridade 2 aumentam o número de folhas de T_i em pelo menos metade do número de vértices adicionados a T_i por elas. □

Lema 3.17. *O Algoritmo 3 é 3-aproximado.*

Prova. Sejam T uma árvore geradora construída pelo Algoritmo 3 e T^* uma árvore geradora com número máximo de folhas. Sabemos que o número máximo de folhas mortas para conectar as árvore de F é igual a $2k$. Assim, o número de folhas de T é pelo menos

a soma do número de folhas de cada árvore T_i menos $2k$. Temos que:

$$\begin{aligned} |V_1(T)| &\geq \sum_{i=0}^{k} |V_1(T_i)| - 2k \\ &\geq (|V(F)| - |B(F)| + 2(k+1))/2 - 2k \\ &= (|V(F)| - |B(F)| - 2k + 2)/2, \end{aligned}$$

onde a segunda desigualdade vem do Lema 3.16.

Assim,

$$\begin{aligned} \frac{|V_1(T^*)|}{|V_1(T)|} &\leq \frac{|V(F)|-2k}{(|V(F)|-|B(F)|-2k+2)/2} \\ &\leq \frac{2(|V(F)|-|B(F)|-2k)+2|B(F)|}{|V(F)|-|B(F)|-2k} \\ &= 2 + \frac{2|B(F)|}{|V(F)|-|B(F)|-2k} \\ &\leq 2 + \frac{2|B(F)|}{4k+3|B(F)|-|B(F)|-2k} \\ &= 2 + \frac{2|B(F)|}{2|B(F)|+2k} \\ &\leq 3, \end{aligned}$$

onde a primeira desigualdade segue do corolário 3.15 e do lema 3.16, a segunda desigualdade multiplicamos por $2/2$, adicionamos $2|B(F)| - 2|B(F)|$ ao numerador, subtraímos 2 do denominador e rearranjamos os termos, a primeira igualdade dividimos $2(|V(F)| - |B(F)| - 2k + 2)$ por $|V(F)| - |B(F)| - 2k + 2$, a terceira desigualdade substituímos $|V(F)|$ por $4k + 3|B(F)|$, pois $|V(F)| \geq \sum_{i=0}^{k} |V(T_i)| \geq 4k + 3|B(F)|$, já que cada árvore tem uma raiz com pelo menos 3 filhos e cada aplicação da expansão de prioridade 1 adiciona 3 vértices, a segunda igualdade rearranjamos os termos, a última desigualdade substituímos $\frac{2|B(F)|}{2|B(F)|+2k}$ por 1. □

Capítulo 4

Algoritmos Aproximados para Grafos Cúbicos

Neste capítulo apresentamos algoritmos aproximados para o PAGMF para grafos cúbicos. Grafos cúbicos são grafos onde todos os vértices têm grau 3. Na primeira seção apresentamos um algoritmo guloso com razão de aproximação 7/4. Na seção seguinte apresentamos um algoritmo 3/2-aproximado que possui a melhor razão de aproximação conhecida para essa classe.

4.1 Algoritmo 7/4-aproximado

Nesta seção apresentamos um algoritmo 7/4-aproximado proposto por Loryś e Zwoźniak [17]. O algoritmo constrói uma floresta F usando algumas regras para adicionar novos vértices. Então conecta as árvores de F e os vértices que não pertencem à floresta para formar uma árvore geradora de G.

4.1.1 Algoritmo

No primeiro passo o algoritmo constroi uma floresta F de G usando três regras. A regra 1 coloca na árvore T_i dois vértices u_1 e u_2 que não pertencem a F adjacente a folha x que pertence a F. Veja ilustração na Figura 4.1 a). A regra 2 coloca na árvore T_i um vértice y que não pertence a F junto com seus dois vizinhos u_1 e u_2 que também não pertencem a F, tal que, y é vizinho de x e x é folha em T_i. Essas regras adicionam duas folhas a árvore e vamos dizer que uma folha está a esquerda e a outra a direita de seu pai. Veja ilustração na Figura 4.1 b). A regra 3 inicia uma nova árvore $T_j, j > i$, enraizada em y e adiciona os vértices y, z, u_1 e u_2, tal que, nenhum deles pertence a F, y tem exatamente um vizinho z que não pertence a F e y, u_1 e u_2 são vizinhos de z. Veja ilustração na Figura 4.1 c).

Definimos prioridades para as regras de expansão onde a regra 1 tem a maior prioridade, a regra 2 tem a segunda maior prioridade e a regra 3 tem a menor prioridade.

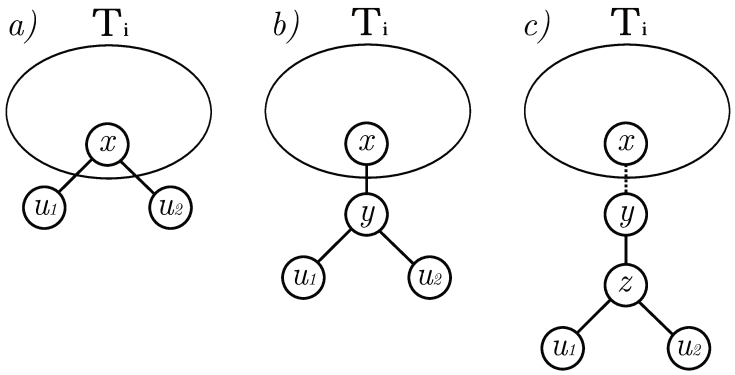


Figura 4.1: Ilustração das regras de expansão: a) regra 1, b) regra 2 e c) regra 3.

No segundo passo o algoritmo adiciona arestas para conectar as árvores de F e os vértices que não pertencem a F.

Entrada: G um grafo cúbico conexo
Saída: T uma árvore geradora de G
início
  $F := \varnothing$;
  $V(T_0) := \{r_0, v_1, v_2, v_3\}$, onde v_1, v_2 e v_3 são vizinhos de um vértice arbitrário r_0 em G;
  enquanto *existe uma regra que pode ser aplicada* **faça**
    Seja i a regra de maior prioridade que pode ser aplicada;
    Seja x a folha de F mais a esquerda que pode ser expandida pela regra i;
    Expanda x;
  fim
  Conecte as árvores em F e todos os vértices que não estão em F para formar uma árvore geradora T de G.
fim

Algoritmo 4: MuitasFolhasCubico74 (G)

Cada regra é aplicada a uma folha no máximo uma vez. Então, o laço é executado no máximo $O(n)$ vezes, onde n é o número de vértices de G. O segundo passo do algoritmo pode ser feito em tempo polinomial. Logo, a complexidade do algoritmo é polinomial.

4.1.2 Definições

Apresentamos algumas definições que usaremos nesta seção. Sejam G um grafo cúbico conexo e $F = \{T_0, \ldots, T_k\}$ uma floresta construída pelo Algoritmo 4. O vértice v é externo se v não pertence a F. Denotamos por EX o conjunto dos vértices externos. Seja R o conjunto de raízes das árvores $T_1, \ldots, T_k$ de F. Denotamos por $\bar{L}(F)$ o conjunto de folhas de F que não pertencem a R. Todo vértice externo tem pelo menos dois vizinhos em uma árvore T_i. Esses vizinhos são chamados de ext-folhas. Uma folha é chamada de livre se não é ext-folha e seus vizinhos não pertencem a R. Denotamos por $\bar{L}_l(F)$ o conjunto de folhas livres de F. Uma aresta uv que pertence a T_i é uma aresta livre se u e v têm grau 3 em T_i. $E_l(T_i)$ denota o conjunto de todas as arestas livres em T_i. Denotamos por $\Gamma_{T_i}(v)$ o conjunto de vértices que são vizinhos de v em T_i.

Apresentamos algumas propriedades da floresta construída pelo Algoritmo 4 que podem ser facilmente verificadas.

Fato 1. Para cada árvore T_i em F existe pelo menos um vértice v de $V(T_i)$ tal que $d_{T_i}(v) = 3$.

Fato 2. Seja $v \in V(T_i)$, $T_i \in F$ e $d_{T_i}(v) = 2$. Se $x \in \Gamma_{T_i}(v)$, então $d(x) = 3$.

Fato 3. Seja $v \in V(T_i)$, $T_i \in F$ e $d_{T_i}(v) = 2$. Se $w \in \Gamma_G(v)$, então $w \in V(T_i)$.

Fato 4. Seja $v \in V(T_i)$, $T_i \in F$ adjacente a um vértice $w \in EX$. Se $u \neq w$ e $u \in \Gamma_G(v)$, então $u \in V(T_i)$.

Fato 5. Seja $v \in V(T_i)$, $T_i \in F$ e $u, w \in \Gamma_G(v)$. Se $u, w \notin V(T_i)$, então $u, w \in V(T_j)$, $j < i$, $T_j \in F$, e T_j é predecessor de T_i.

Fato 6. Seja $v \in V(G)$ e $u, w, z \in \Gamma_G(v)$. Se $v \in EX$, então dois de seus vizinhos, digamos u, w, pertencem a $\bar{L}(T_i)$ para alguma árvore $T_i \in F$, e o terceiro vizinho z satisfaz uma das seguintes condições:

- $z \in \bar{L}(T_i)$;
- $z \in \bar{L}(T_j)$, $j > i$, $T_j \in F$, T_i é predecessor de T_j;
- $z \in EX$ e seus vizinhos $z_1, z_2 \in \bar{L}(T_j)$, $T_j \in F$, onde $i = j$, ou T_i é um predecessor de T_j, ou T_j é um predecessor de T_i.

4.1.3 Garantia da razão de aproximação

Nesta subseção provamos que o Algoritmo 4 é 7/4-aproximado apresentando o número de folhas em uma solução ótima e um limite inferior para o número de folhas para uma árvore produzida pelo algoritmo.

O seguinte lema apresenta o número de folhas em uma árvore geradora em termos do número de vértices com grau 2 na árvore.

Lema 4.1. *Seja G um grafo cúbico com pelo menos 4 vértices e seja T^* uma árvore*

geradora com número máximo de folhas de G. Se d vértices tem grau 2 em T^*, então $|V_1(T^*)| = (|V(G)| - d + 2)/2$.

Prova. Seja T um árvore geradora de G. Seja v um vértice de grau 2 em T e u e w seus vizinhos. Chamaremos a remoção de v e a adição da aresta uw em T de substituir v por uma aresta. Seja T' a árvore construída a partir de T após a substituição de todo v de grau 2 por uma aresta. Claramente o número de folhas de T e T' é o mesmo. Além disso, T' é uma árvore. Logo:

$$\begin{aligned} 3|V_3(T')| + |V_1(T')| &= 2(|V_3(T')| + |V_1(T')| - 1) \\ |V_1(T')| &= (|V_3(T')| + |V_1(T')| + 2)/2 \\ |V_1(T)| &= (|V(G)| - d + 2)/2, \end{aligned}$$

onde a primeira igualdade vem do fato de que a soma dos graus dos vértices de uma árvore é igual a duas vezes o número de arestas, a segunda igualdade apenas rearranjamos os termos, e a terceira igualdade substituímos $|V_1(T')|$ por $|V_1(T)|$ do lado esquerdo e $|V_3(T')| + |V_1(T')|$ por $|V(G)| - d$ do lado direito, onde d é o número de vértices de grau 2 em T. Isso vale para qualquer árvore T, logo vale pra uma árvore com número máximo de folhas. □

O próximo lema e corolário garantem que pelo menos um terço dos vértices em uma floresta F são folhas.

Lema 4.2. *Seja T uma árvore de F, onde F é a floresta construída pelo Algoritmo 4 para um grafo cúbico G, com pelo menos 4 vértices. Então $|V(T)| = 3|V_1(T)| - |E_l(T)| - 5$.*

Prova. Sejam G um grafo cúbico com pelo menos 4 vértices, F uma floresta construída pelo Algoritmo 4 e T uma árvore de F. Pelo fato 2 todo vértice v com grau 2 em T é adjacente a dois vértices, digamos u e w, de grau 3. Vamos excluir de T o vértice v e as arestas incidentes a ele e adicionar uma nova aresta uw. Observe que o número de folhas da árvore resultante não é alterado e que os graus de u e w não são alterados. Ao executar o processo para todo vértice v com grau 2 em T temos uma árvore resultante T'. Seja d o número de arestas que foram adicionas a T', ou seja, $d = |V_2(T)|$ já que para cada vértice excluído de T uma aresta foi adicionada. Sabemos que $|E(T')| = |V(T)| - d - 1$. Temos:

$$\begin{aligned} |V(T)| - d - 1 &= d + |V_1(T)| + |E_l(T)| \\ d &= (|V(T)| - 1 - |V_1(T)| - |E_l(T)|)/2. \end{aligned}$$

Além disso,

$$\begin{aligned}
|V_1(T)| &= |V_1(T')| \\
|V_1(T)| &= (|V(T)| - d + 2)/2 \\
2|V_1(T)| &= |V(T)| - d + 2 \\
2|V_1(T)| &= |V(T)| - ((|V(T)| - 1 - |V_1(T)| - |E_l(T)|)/2) + 2 \\
4|V_1(T)| &= 2|V(T)| - (|V(T)| - 1 - |V_1(T)| - |E_l(T)|) + 4 \\
|V(T)| &= 3|V_1(T)| - |E_l(T)| - 5,
\end{aligned}$$

onde a primeira igualdade é válida porque para construir T' o número de vértices não é alterado, a segunda igualdade vem do lema 4.1, a terceira igualdade multiplicamos ambos os lados por 2, a quarta substituímos d por $(|V(T)| - 1 - |V_1(T)| - |E_l(T)|)/2$, a quinta multiplicamos ambos os lados por 2 e a sexta igualdade apenas rearranjamos os termos. □

Corolário 4.3. *Seja F a floresta construída pelo Algoritmo 4 para um grafo cúbico G, com pelo menos 4 vértices. Se F tem $k+1$ árvores disjuntas, então $|V(F)| = 3|V_1(F)| - |E_l(F)| - 5k - 5$.*

O lema seguinte é usado na prova da razão de aproximação do Algoritmo 4. Ele diz que o número de árvores em F pode ser limitado pela combinação de arestas livres, folhas livres e o número de vértices de grau 2 em uma solução ótima.

Lema 4.4. *Sejam G um grafo cúbico e T^* uma árvore geradora com número máximo de folhas de G. Seja F uma floresta construída pelo Algoritmo 4. Então $2|E_l(F)| + |\bar{L}_l(F)| + 2d + 6 > k$, onde d é o número de vértice de grau 2 em T^* e $k+1$ é o número de árvores em F.*

A prova deste lema é muito técnica e resolvemos omiti-la. Apresentamos a seguir a ideia da prova apresentada por Loryś e Zwoźniak [17]. Primeiro são atribuídos pesos para arestas livres, folhas livres e vértices de grau 2 em uma solução ótima de forma que a soma desses pesos é no máximo $2|E_l(F)| + |\bar{L}_l(F)| + 2d + 6$. Em seguida os pesos são divididos e parte deles são movidos para algumas folhas de F. Finalmente, mostram que cada par de folhas adjacentes as raízes de R recebeu pelo menos peso 1 e que a soma dos pesos movidos foi maior do que k.

Teorema 4.5. *Seja T^* uma árvore geradora com número máximo de folhas para um grafo G com pelo menos 4 vértices. Seja T uma árvore geradora construída pelo Algoritmo 4 para G. Então $\frac{|V_1(T^*)|}{|V_1(T)|} < \frac{7}{4}$.*

Prova. Sejam G um grafo cúbico, F e T a floresta e árvore construídas pelo algoritmo e T^* uma solução ótima de G. Sabemos que $|V_1(T)| = |V_1(F)| - 2k$, onde $k+1$ é

o número de árvores de F, pela construção do algoritmo. Pelo lema 4.1 sabemos que $|V_1(T^*)| = (|V(G)| - d + 2)/2$, onde d vértices tem grau 2 em T^*.

$$\begin{aligned}
\frac{|V_1(T^*)|}{|V_1(T)|} &= \frac{(|V(G)|-d+2)/2}{|V_1(F)|-2k} \\
&= \frac{1}{2} * \frac{3|V_1(F)|-5(k+1)-|E_l(F)|+|EX|-d+2}{|V_1(F)|-2k} \\
&\leq \frac{1}{2} * \frac{3|V_1(F)|-5(k+1)-|E_l(F)|+(|V_1(F)|-|\bar{L}_l(F)|-3k)/2-d+2}{|V_1(F)|-2k} \\
&= \frac{1}{4} * \frac{7|V_1(F)|-13k-2|E_l(F)|-|\bar{L}_l(F)|-2d-6}{|V_1(F)|-2k} \\
&\leq \frac{1}{4} * \frac{7|V_1(F)|-13k-k}{|V_1(F)|-2k} \\
&\leq \frac{7}{4},
\end{aligned}$$

onde, a segunda igualdade vem da definição de EX e do corolário 4.3, a primeira desigualdade vem da definição de ext-folhas, folhas livres e do fato 5 e a segunda desigualdade vem do lema 4.4. □

4.2 Algoritmo 3/2-aproximado

Nesta seção apresentamos um algoritmo 3/2-aproximado proposto por Bonsma e Zickfeld em [3]. O algoritmo trabalha com conjuntos conexos dominantes ao invés de construir uma árvore diretamente. Ele começa com um conjunto conexo dominante que é o conjunto de todos os vértices do grafo, e retira vértices desse conjunto mantendo a conexidade e dominância até obter um conjunto minimal. Os vértices pertencentes a este conjunto serão vértices internos da árvore geradora e os vértices restantes serão folhas.

A seguir apresentamos algumas definições que usaremos nesta seção, seguido do algoritmo e da prova da razão de aproximação.

4.2.1 Definições

A seguir apresentamos algumas definições que usaremos nesta seção. Um triângulo é um k_3. Um diamante é um k_4 menos uma aresta. Seja H um subgrafo de G, definimos $Int(H)$ como o conjunto de vértices que só tem vizinhos em H. Um CD-set é um conjunto de vértices de G que é dominante e conexo. Usaremos n para indicar o número de vértices em G. Seja G um grafo cúbico conexo. Um CD-set S é chamado de um 2-CD-set se todo vértice em $\bar{S}$ tem no máximo dois vizinhos em S. Um conjunto é dito minimal para alguma propriedade sempre que não existe um subjconjunto estrito com essa propriedade.

Para um grafo G cúbico, seja $V^{\Delta}(G) \subseteq V(G)$ o conjunto de vértices de G que fazem parte de triângulos. Dizemos que um triângulo ou diamante H de G é do tipo i se ele tem exatamente i vizinhos que não fazem parte de triângulos ou diamantes. Denotamos por $T_i(G)$ ($D_i(G)$) o número de triângulos (diamantes) do tipo i em G.

Uma aresta uv é um T-ponte se u e v pertencem a triâgulos ou diamantes, mas não pertencem ao mesmo triângulo ou diamante. O grafo $f_1(G)$ é obtido apagando todos os vértices de G que pertencem a triângulos ou diamantes do tipo 0 e apagando também todas as T-pontes. O grafo $f_2(G)$ é obtido apagando todos os vértices que pertencem a triângulos ou diamantes. Sejam $cc_1(G) = cc(f_1(G)) - cc(G)$, e $cc_2(G) = cc(f_2(G)) - cc(f_1(G))$, onde $cc(G)$ é igual ao número de componentes conexas de G. Agora podemos definir o parâmetro $x(G)$ como $x(G) = 2cc_1(G) + cc_2(G) + D_0(G) + T_0(G)$.

4.2.2 Algoritmo

Apresentamos o Algoritmo 5 que recebe como entrada um grafo cúbico conexo e retorna um 2-CD-set minimal S, e um CD-set minimal S'. A seguir apresentamos o algoritmo e uma análise de sua complexidade de tempo.

Entrada: G um grafo cúbico conexo
Saída: Um 2-CD-set minimal S, e um CD-set minimal $S' \subseteq S$

```
1  início
2      Fase 1:
3      S_1 := V(G).
4      enquanto ∃ T-ponte uv em G[S_1] tal que S_1 − u − v é um CD-set de G faça
5          S_1 := S_1 − u − v.
6      fim
7      Fase 2:
8      S_2 := S_1.
9      Seja T um triângulo do tipo 3 com V(T) ⊆ S_2.
10     enquanto ∃ T tal que S_2\V(T) é um CD-set de G faça
11         S_2 := S_2\V(T).
12     fim
13     Seja D um diamante do tipo 2 com V(D) ⊆ S_2.
14     enquanto ∃ D tal que S_2\Int(D) é um CD-set de G faça
15         S_2 := S_2\Int(D).
16     fim
17     Fase 3:
18     S := S_2.
19     Remova vértices de S até que S seja um 2-CD-set minimal de G.
20     S' := S.
21     Remova vértices de S' até que S' seja um CD-set minimal de G.
22 fim
```

Algoritmo 5: PequenoCD-set32 (G)

O algoritmo começa com um conjunto S_1, que inicialmente é igual ao conjunto de vértices do grafo. Na fase 1 ele retira iterativamente vértices u e v de S_1, tal que, uv é uma T-ponte de $G[S_1]$ e $S_1 - u - v$ ainda é um CD-set. Na fase 2 ele retira vértices de triângulos do tipo 3 e diamantes do tipo 2, de modo que o conjunto resultante mantenha-se dominante e conexo. Na última fase remove vértices para encontrar os conjuntos minimais.

O Algoritmo 5 é polinomial já que cada passo é polinomial e o tamanho do conjunto dominante nunca cresce. Além disso, as mudanças feitas em S_1 e S_2 nas fases 1 e 2 sempre removem conjunto de vértices adjacentes, logo, eles terão no máximo dois vizinhos no CD-set resultante. A propriedade de ser um 2-CD-set é mantida sempre, e assim, S será um 2-CD-set minimal.

Tendo um conjunto conexo dominante S de G podemos encontrar uma árvore geradora da seguinte maneira. Os vértices de S serão vértices internos da árvore e os outros vértices serão folhas. Sabemos que S é conexo. Então podemos encontrar uma árvore geradora em $G[S]$ e em seguida adicionar arestas para conectar os vértices de G que não pertencem a S.

4.2.3 Garantia da razão de aproximação

Nesta seção provamos que a razão de aproximação do Algoritmo 5 é 3/2. Seja G um grafo cúbico, vamos mostrar que uma árvore construída pelo algoritmo tem pelo menos $(n - x(G) + 4)/3$ folhas e que uma árvore geradora qualquer de G não tem mais do que $(n - x(G) + 2)/2$ folhas.

Um subgrafo H de G é um bloco de G se é um subgrafo 2-conexo maximal. Isso significa que pontes de G não estão em nenhum bloco. Seja S um CD-set de G. Se $S - v$ não é um conjunto dominante de G, então v é dominador de S. Se $G[S - v]$ não é conexo, então v é conector de S.

Teorema 4.6. *Seja G um grafo cúbico conexo. Sejam S um 2-CD-set minimal de G onde $G[S]$ tem b^{Δ} blocos que contêm triângulos, e $S' \subseteq S$ um CD-set minimal de G. Então $|S'| \leq (2n + b^{\Delta}(S') - 4)/3$.*

Antes de apresentar uma prova do teorema 4.6 enunciamos dois lemas que serão usados na prova.

Lema 4.7. *Seja S um 2-CD-set minimal de um grafo cúbico G com um vértice $v \in S$ que não é conector nem dominador, que é parte de bloco $G[B]$ de $G[S]$. Para qualquer CD-set $S' \subseteq S$ com $B \subseteq S'$, v não é conector nem dominador de S'.*

Prova. O conjunto S é um 2-CD-set minimal. Então v tem três vizinhos em S. Como v não é conector todos os seus vizinhos estão em B. Portanto, independente de quais

vértices foram removidos para obter S', enquanto $B \subseteq S'$ v não será dominador nem conector. □

Lema 4.8. *Seja S um 2-CD-set minimal de um grafo cúbico G. Para um bloco H de $G[S]$, seja $D_i \in V(H)$ os vértices que têm i vizinhos em H, $i \in \{2, 3\}$. Então $|D_2| \geq |D_3|$, e se H não contém triângulos, $|D_2| \geq |D_3| + 1$.*

Prova. Apresentamos uma prova por casos. Considere H um bloco qualquer de $G[S]$. Queremos mostrar que $|D_2| \geq |D_3|$.

Primeiro vamos mostrar que todo vértice em D_3 tem no máximo um vizinho em D_3. Seja u um vértice em D_3. Suponha por contradição que u tem dois vizinhos, v e w, em D_3. Seja $S' = S - u - v$. Note que S' é dominante e que todo vértice em $\bar{S}'$ tem no máximo 2 vizinhos em S'. Então S' não é conexo, já que S é 2-CD-set minimal. Pelo mesmo raciocínio $S'' = S - u - w$ também não é conexo.

Sabemos que u e v têm exatamente um vizinho em cada componente de S', já que eles não são vértices de corte de S. O vértice w também não é vértice de corte em S. Então v e w não são vizinhos. Pelo mesmo motivo u e v não têm vizinhos em comum de grau 3. Como $S - w$ é conexo, temos que $S - u - w$ também é conexo. Mas isso é uma contradição ao fato da conexidade de S''. Então todo vértice em D_3 tem no mínimo dois vizinhos em D_2. Logo, $|D_2| \geq |D_3|$.

Agora considere o caso que H não tem triângulo. Queremos mostrar que $|D_2| \geq |D_3| + 1$. Como provamos o caso anterior, agora basta mostrar que se $|D_2| = |D_3|$, então H tem um triângulo.

Seja H um bloco de S tal que $|D_2| = |D_3|$. Então todo vértice em D_3 tem exatamente um vizinho em D_3 e todo vértice em D_2 tem dois vizinhos em D_3. Queremos mostrar que H tem um triângulo.

Sejam dois vizinhos $u, v \in D_3$. Já vimos que $H - u - v$ não é conexo. Sejam C_1 e C_2 os componentes de $H - u - v$, tal que $|V(C_1)| \leq |V(C_2)|$. Se C_1 contém vértices de D_3, então sejam u' e v' dois vizinhos em $H \cap V(C_1)$. Sejam C_1' e C_2' os componentes de $H - u' - v'$, tal que $\{u, v\} \subset V(C_2')$. Note que $V(C_2) \subset V(C_2')$ e $|V(C_1')| < |V(C_1)|$. Podemos repetir esse processo até que o menor componente não tenha vértices de D_3. Como todo vértice em D_2 tem dois vizinhos em D_3, este componente tem só um vértice, e encontramos um triângulo.

Mostramos que se $|D_2| = |D_3|$, então H tem um triângulo. Logo, se H não tem triângulo, então pelo primeiro caso sabemos que $|D_2| \geq |D_3| + 1$. □

Prova do Teorema 4.6. Sejam L_i o conjunto dos vértices em $\bar{S}$ que têm i vizinhos em S $(i = 1, 2)$, e $L_3 = S \backslash S'$. Então temos

$$\bar{S}' = L_1 \cup L_2 \cup L_3.$$

Considere G' o grafo obtido apagando L_2, e apagando todas as arestas que são incidentes a dois vértices em L_1. Considere a árvore T obtida de G' pela contração dos blocos para um vértice. Denotamos por $V_C \subset V(T)$ os vértices de T obtidos de G' pela contração.

Particionamos S' nos conjuntos X, N e V_B definidos a seguir. Denotamos por X o conjunto de vértices v de G' tal que $G' - v$ tem três componentes. O conjunto N é o conjunto de vértices de grau 2 em G'. Chamamos de V_B o conjunto de vértices que estão em um bloco de G', excluindo vértices em L_3.

Para todo bloco $G'[B]$ de G', particionamos B em D_2 e D_3, os vértices de grau 2 e 3 respectivamente em $G'[B]$. Seja $I^\Delta = 1$ se $G'[B]$ contém um triângulo, e $I^\Delta = 0$ caso contrário. Pelo lema 4.8 sabemos que $|D_2| \geq |D_3| + 1 - I^\Delta(B)$. Seja $L_3^B = L_3 \cap B$. Como G é cúbico, um vértice em D_3 não pode ser conector ou dominador já que ele está em um bloco e seus vizinhos também. Se $D_3 \neq \emptyset$, então pelo lema 4.7 B tem pelo menos um vértice de L_3, e $|L_3^B| \geq 1$. Se $D_3 = \emptyset$, então $G'[B]$ tem um ciclo, e temos $|D_2| \geq 4 - I^\Delta(B)$. Em ambos os casos a seguinte desigualdade é válida

$$3|L_3^B| + 2|D_2| \geq |D_2| + |D_3| + 4 - I^\Delta(B).$$

Seja $v_B \in V_C$ o vértice de T correspondente a B. Um vértice x em D_2 tem um vizinho em G' que não está no bloco $G'[B]$, pois x tem dois vizinhos em B e o terceiro não poderia pertencer a L_2 já que x não é conector. Então $d_T(v_B) \geq |D_2|$. Substituindo na desigualdade anterior temos

$$3|L_3^B| + 2d_T(v_B) \geq |B| + 4 - I^\Delta(B)$$

$$2|L_3^B| + 2(d_T(v_B) - 2) \geq |B \backslash L_3^B| - I^\Delta(B).$$

Somando para todos os blocos de G':

$$2|L_3| + 2\sum_{v \in V_C} (d_T(v) - 2) \geq |V_B| - b^\Delta$$

$$|L_3| \geq |V_B|/2 - b^\Delta/2 - \sum_{v \in V_C} (d_T(v) - 2).$$

Por definição os vértices em N são adjacentes a vértices em L_2. Cada vértice de L_2 tem no máximo dois vizinhos em N. Então, $|L_2| \geq |N|/2$. Como T é uma árvore, sabemos que $\sum_{v \in V(T)} d(v) = 2(n(T) - 1)$, que nos dá $|L_1| \geq \sum_{v \in V(T) \backslash L_1} (d_T(v) - 2) + 2$. Combinandos limites inferiores:

$$\bar{S}' = L_1 \cup L_2 \cup L_3 \geq$$

$$|V_B|/2 - b^\Delta/2 - \sum_{v \in V_C} (d_T(v) - 2) + |N|/2 + \sum_{v \in V(T) \backslash L_1} (d_T(v) - 2) + 2 =$$

$$(|V_B| - b^{\Delta} + |N|)/2 + \sum_{v \in V(T) \setminus L_1 \setminus V_C} (d_T(v) - 2) + 2.$$

Sabemos que $V(T) \setminus L_1 \setminus V_C = X \cup N$, e que vértices de N têm grau 2. Logo,

$$\bar{S'} \geq (|V_B| - b^{\Delta} + |N|)/2 + |X| + 2 \geq$$

$$|S'|/2 - b^{\Delta}/2 + |X|/2 + 2 \geq |S'|/2 - b^{\Delta}/2 + 2,$$

onde na segunda desigualdade usamos $|S'| = |V_B| + |N| + |X|$. Adicionando $|S'|$ dos dois lados e rearranjando os termos temos finalmente o resultado que queríamos demonstrar:

$$|S'| \leq (2n(G) + b^{\Delta} - 4)/3.$$

□

Teorema 4.9. *Seja S um 2-CD-set minimal de G construído pelo Algoritmo 5. Então, o número de triângulos mais o número de diamantes em $G[S]$ é no máximo $2cc_1(G) + T_0(G) + D_0(G) + cc_2(G)$.*

Prova. Vamos dividir a prova em duas partes. Primeiro vamos mostrar que o número de triângulos dos tipos 0, 1 e 2 mais o número de diamantes dos tipos 0 e 1 são no máximo $2cc_1(G) + T_0(G) + D_0(G)$. Depois basta mostrar que o número de triângulos do tipo 3 mais o número de diamantes do tipo 2 são no máximo $cc_2(G)$.

Apresentamos duas definições e em seguida uma afirmação que usaremos nesta prova. Uma T-ponte uv de G é chamada de S_1-ponte se u e v pertencem a S_1. Denotamos por $n_B(S_1)$ o número de S_1-pontes em $G[S_1]$.

Afirmação 1. Uma S_1-ponte uv é uma ponte de $G[S_1]$.

Prova. Seja uv uma S_1-ponte. Suponha que uv não é ponte de $G[S_1]$. Já que u e v fazem parte de triângulo ou diamante, podemos remover u e v de S_1 e o grafo $G[S_1]$ será conexo. Mas $S_1 - u - v$ não é dominante, porque u e v não foram removidos de S_1 na execução do algoritmo. O que destroi a propriedade de ser dominante é que todos os vizinhos de um dos vértices, digamos u, já foram removidos de S_1. Mas u seria folha de $G[S_1]$ e uv seria ponte, contradição. Então uv é ponte de $G[S_1]$. □

Seja G' o subgrafo de $G[S_1]$ obtido após a remoção de todas as S_1-pontes. Os componentes de G' são de dois tipos: aqueles que fazem parte de componentes de $f_1(G)$, e aqueles que fazem parte de triângulos e diamantes do tipo 0. Pela maneira que os vértices são removidos de S_1 na fase 1 do algoritmo, todo componente de $f_1(G)$ e todo triângulo ou diamante do tipo 0, tem pelo menos um vértice em S_1. Já que todas as S_1-pontes

são pontes de $G[S_1]$ (afirmação 1), contraindo toda aresta de G' que não é uma S_1-ponte temos uma árvore. Logo, sabemos que o número de S_1-pontes é

$$n_B(S_1) = cc_1 + T_0 + D_0.$$

Seja $T_{i,j}$ ($D_{i,j}$) o número de triângulos (diamantes) do tipo i de G que contém j vértices de S_1. Contando as S_1-pontes incidentes aos triângulos e diamantes temos:

$$3T_{0,3} + 2T_{1,3} + T_{2,3} + T_{1,2} + T_{0,1} + 2T_{0,2} + 2D_{0,4} + D_{1,4} + D_{0,3} \leq 2n_B(S_1).$$

Desenvolvendo a inequação anterior, encontramos um limitante para o número de triângulos dos tipos 0, 1 e 2 e de diamantes dos tipos 0 e 1 de G que todos os seus vértices pertencem a $G[S_1]$:

$$\begin{gathered} T_{0,3} + T_{1,3} + T_{2,3} + D_{0,4} + D_{1,4} \leq \\ 2n_B(S_1) - 2T_{0,3} - T_{1,3} - T_{1,2} - T_{0,1} - 2T_{0,2} - D_{0,4} - D_{0,3} = \\ 2(cc_1 + T_0 + D_0) - 2T_{0,3} - T_{1,3} - T_{1,2} - T_{0,1} - 2T_{0,2} - D_{0,4} - D_{0,3} = \\ 2cc_1 + T_0 + D_0 - T_{0,3} - T_{1,3} - T_{1,2} - T_{0,2} \leq 2cc_1 + T_0 + D_0. \end{gathered}$$

Agora analisamos a fase 2 do algoritmo. Seja S_2 o conjunto de vértices S_2 logo após o término da fase 2 do algoritmo. Sejam S_2-triângulos os triângulos T do tipo 3 com $V(T) \subseteq S_2$ e S_2-diamantes os diamantes D do tipo 2 com $V(D) \subseteq S_2$. Vamos mostrar que o número de S_2-triângulos mais o número de S_2-diamantes é no máximo cc_2.

Sejam G um grafo cúbico e C um componente de $f_1(G)$. Por definição, sabemos que $cc(f_2(C)) = cc_2(C) + 1$. Primeiro vamos mostrar que o número de S_2-triângulos mais o número de S_2-diamantes em C é no máximo $cc_2(C)$.

Seja T um S_2-triângulo ou S_2-diamante em C. Sabemos que $G[S_2 \backslash V(T)]$ é dominante porque todos os vizinhos de T estão em S_2. Além disso, $V(T)$ não foi removido na fase 2, logo $G[S_2 \backslash V(T)]$ é desconexo. Então, $V(T)$ contém pelo menos um vértice de corte de C.

Considere o grafo G' definido a seguir. Para todo S_2-triângulo e todo S_2-diamante, adicionamos um vértice preto a G', e para todo componente de $f_2(G)$ em C nós adicionamos um vértice branco em G'. Para todo S_2-triângulo ou S_2-diamante adjacente a um componente de $f_2(G)$, nós adicionamos uma aresta entre os vértices correspondentes. Isso nos dá um grafo G' bipartido com $cc_2(C) + 1$ vértices brancos. Note que todo vértice preto é vértice de corte em G'. Logo, o número de vértices pretos de G' é no máximo o número de vértices brancos menos um, ou seja, é no máximo $cc_2(C)$.

Todo S_2-triângulo e S_2-diamante de G estão em algum componente C de $f_1(G)$. Se Y é o conjunto de componentes de $f_1(G)$, então $cc_2(G) = \sum_{C \in Y} cc_2(C)$. Segue que o número de triângulos do tipo 3 mais o número de diamantes do tipo 2 em S_2 é no máximo $cc_2(G)$.

Já que S é um subconjunto de S_1 e de S_2, sabemos que o número de triângulos e diamantes que pertencem a $G[S]$ também são limitados por $2cc_1 + T_0 + D_0 + cc_2$. □

A seguir vamos apresentar um limite superior para o número de folhas de uma árvore geradora de um grafo cúbico. Primeiro observe que só precisamos provar isso para árvores da seguinte forma.

Lema 4.10. *Seja G um grafo cúbico conexo. Existe uma árvore geradora com número máximo de folhas T de G que contém duas arestas de todos os triângulos dos tipos 0, 1 e 2 de G e contém ou zero ou duas arestas de todo triângulo do tipo 3 de G.*

Prova. Seja T uma árvore geradora com número máximo de folhas de G. Vamos monstrar que o lema 4.10 é verdadeiro fazendo alterações em T sem reduzir o número de folhas.

Sejam H um triângulo de G e $V(H) = \{v_1, v_2, v_3\}$. Se T contém apenas uma aresta de H, digamos v_1v_2, então, sabemos que v_3 é uma folha e que pelo menos de v_1 e v_2, digamos v_1, não é folha. Assim, removemos a aresta incidente a v_3 e adicionamos a aresta v_1v_3. Dessa forma a quantidade de folhas não diminui.

Agora considere o caso que T não contém aresta de H. Então os vértices de H são folhas de T. Podemos assumir que H não é do tipo 3, logo H é adjacente a algum triângulo ou diamante H'. Sem perca de generalidade vamos assumir que H' é adjacente a v_1. Seja $u \in V(H')$ o vértice adjacente a v_1, tal que, uv_1 pertence a T. Sabemos que u tem grau pelo menos 2. Então podemos fazer as seguintes trocas de arestas. Adicionamos as arestas v_1v_3 e v_2v_3 e removemos as arestas uv_1 e v_2a_2, onde a_2 é o vértice adjacente a v_2 que não pertence a H. Após a troca v_3 deixa de ser folha, mas u passa a ser folha.

Essas operações não diminuem o número de folhas. Então podemos aplicar essas mudanças até que a propriedade seja válida. □

Teorema 4.11. *Seja T uma árvore geradora de um grafo cúbico G. Então T tem no máximo $(n - x(G) + 2)/2$ folhas.*

Prova. Para árvores com grau máximo 3, sabemos que o número de vértices com grau 3 é igual ao número de vértices com grau 1 menos 2, ou seja, $|V_3(T)| = |V_1(T)| - 2$, onde T é uma árvore geradora de um grafo cúbico. Além disso, sabemos que a soma dos graus dos vértices de T é igual a duas vezes o número de arestas e que o número de arestas é igual ao número de vértices menos um. Logo,

$$\begin{aligned} 3|V_3(T)| + 2|V_2(T)| + |V_1(T)| &= 2(|V(T)| - 1) \\ &= 2(|V_3(T)| + |V_2(T)| + |V_1(T)| - 1) \\ 3|V_3(T)| + |V_1(T)| &= 2(|V_1(T)| + |V_3(T)|) \\ |V_3(T)| &= |V_1(T)| - 2 \end{aligned}$$

Assim,

$$\begin{aligned} n &= |V_1(T)| + |V_2(T)| + |V_3(T)| \\ &= |V_1(T)| + |V_2(T)| + |V_1(T)| - 2 \\ 2|V_1(T)| &= n - |V_2(T)| + 2 \\ |V_1(T)| &= (n - |V_2(T)| + 2)/2. \end{aligned}$$

Para provar o teorema 4.11 basta mostrar que $x(G) \leq |V_2(T)|$. Para isso vamos definir G', T' e T''.

Seja G' um grafo obtido de G pela contração de todo triângulo e diamante para um vértice preto, e pela contração de todo componente de $f_2(G)$ para um vértice branco. Denotamos por P o conjunto de vértices pretos e por B o conjunto de vértices brancos de G'. As arestas de G que não foram contraídas na construção de G' são chamadas de G'-arestas.

O subgrafo gerador T' de G' é construído da seguinte maneira. Uma aresta de G' é adicionada a T' se a aresta correspondente e em G pertence a T e se e não é incidente a uma folha de T que faz parte de triângulo ou diamante. Triângulos do tipo 3 que contêm três folhas de T vão corresponder a um vértice isolado v em T' neste ponto. Para garantir que T' seja conexo, adicionamos uma aresta arbitrária de G' que seja adjacente a v de T'.

Afirmação 2. T' é conexo.

Prova. Sejam u e v vértices distintos de G'. Cada um deles representa um subgrafo de G, digamos H_u e H_v de G respectivamente. Sejam x_u um vértice de H_u e x_v um vértice de H_v. Queremos mostrar que existe um caminho entre x_u e x_v para quaisquer H_u e H_v distintos. Sabemos que existe um caminho P de x_u a x_v em T, já que T é conexo. Toda G'-aresta em P corresponde a uma aresta em T'. Para a primeira e última arestas isso não é trivial, mas se escolhermos x_u e x_v de maneira adequada podemos garantir isso. O vértice x_u que pertence a H_u de ser escolhido como a seguir. Se H_u é um triângulo do tipo 0, 1 ou 2 ou se H_u é um componente de $f_2(G)$, então escolha x_u de forma arbitrária. Se H_u é um triângulo do tipo 3, então adicionamos a aresta uw a $E(T')$, para algum w em $V(G')$. Seja x_u o vértice incidente a esta aresta. Se H_u é um diamante, então escolha x_u como um vértice que não é folha em T. O vértice x_v é escolhido da mesma forma. Pela construção de T', escolha de x_u e x_v e pelo lema 4.10, o caminho P corresponde a um caminho de u a v em T'. Então T' é conexo, já que existe um caminho entre quaisquer dois vértices de T'. □

Afirmação 3. Se um vértice preto v tem grau i em T', então o correspondente subgrafo H_v de G contém pelo menos $i - 1$ vértices que tem grau 2 em T.

Prova. O subgrafo H_v pode ser um diamante ou um triângulo. Se H_v é um diamante, então só precisamos verificar o caso em que $i = 2$. Pela construção de T' os dois vértices

de H_v adjacentes a G'-arestas não são folhas, e pelo menos um deles não tem grau 3, senão haveria um ciclo. Então pelo menos um vértice tem grau 2. Se H_v é um triângulo, então ou T contém duas arestas de H_v, ou H_v é do tipo 3 que contém três folhas de T, mas nesse último caso v teria grau 1 e a prova é trivial. Suponha que H_v contém duas arestas de T. Se $i = 1$ a prova é trivial, então vamos verificar os outros dois casos. Se $i = 2$, então H_v tem um ou três vértices de grau 2 em T. Se $i = 3$, então H_v tem dois vértices de grau 2 em T. Então, H_v de G tem pelo menos $i - 1$ vértices que tem grau 2 em T. □

Pela afirmação 2 sabemos que T' é conexo. Então,

$$|E(T')| \geq |V(T')| - 1 = |P| + |B| - 1.$$

Seja T'' o subgrafo gerador de T' que contém as arestas de T' que são incidentes a pelo menos um vértice branco. Observe que T'' não tem T-pontes e que $cc(T'') = D_0(G) + T_0(G) + cc_1 + 1$. Quando adicionamos arestas de T' uma por uma até que T'' é obtida, claramento a adição de cada aresta pode diminuir o número de componentes conexas em no máximo um. Assim, $cc(T'') - 1$ é o número mínimo de arestas de T' que são incidentes a dois vértices pretos. Então,

$$\begin{aligned} \textstyle\sum_{v \in P}(d_{T'}(v) - 1) &\geq |E(T')| + (cc(T'') - 1) - |P| \\ &\geq |P| + |B| - 1 + D_0 + T_0 + cc_1 - |B| \\ &= 2cc_1 + cc_2 + D_0 + T_0 \\ &= x(G), \end{aligned}$$

onde a primeira desigualdade vem do fato da soma dos graus dos vértices pretos serem pelo menos o número de arestas de G' mais o número de arestas de G' incidentes a dois vértices pretos, para a igualdade usamos $|B| = cc_1 + cc_2 + 1$. Já que os vértices de T' com grau 2 contam pelo menos um vértice de grau 2 em T, e, pela afirmação 3, vértices de grau 3 contam pelo menos dois, $x(G)$ também é um limitante inferior para o número de vértices com grau 2 em T, ou seja, $x(G) \leq |V_2(T)|$. □

Então,

$$\begin{aligned} \frac{|V_1(T*)|}{|V_1(T)|} &\leq \frac{(n - x(G) + 2)/2}{(n - x(G) + 4)/3} \\ &\leq \frac{3}{2}, \end{aligned}$$

onde T^* é uma árvore geradora qualquer e T é uma árvore geradora devolvida pelo Algoritmo 5.

Capítulo 5

Algoritmo Exato e Experimentos Computacionais

Neste capítulo apresentamos um algoritmo exato para o PAGMF baseados em Programação Linear Inteira e os resultados dos experimentos computacionais feitos para os algoritmos apresentados neste trabalho.

5.1 Algoritmo Exato

Lucena, Maculan e Simonetti em [20] propõem duas formulações para o PAGMF: a primeira é baseada em uma versão da literatura para grafos orientados; a segunda o formula como um problema da arborescência de Steiner. Ambos são resolvidos usando uma estratégia de *Branch-and-Cut*. Além disso implementaram uma versão melhorada do algoritmo proposto por Fujie em [9] e fizeram testes com os três métodos usando como entrada grafos gerados aleatoriamente. A formulação para grafos orientados obteve os melhores resultados em média e é apresentada a seguir.

5.1.1 Uma formulação para grafos orientados

Apresentamos uma formulação do PAGMF como um problema de encontrar uma arborescência geradora com muitas folhas. Sejam $G = (V, E)$ um grafo e $r \in V$ um vértice arbitrário. Construa o grafo orientado $D = (V, A)$, obtido de G, com $A = \{(i, j), (j, i) : e = [i, j] \in E, i, j \in V \backslash \{r\}\} \cup \{(r, j) : [r, j] \in E\}$. Em relação ao grafo D, para $i \in V$, denotamos por $\delta^+(i)$ o conjunto $\{(i, j) \in A : j \in V\}$ e dizemos que as arestas de $\delta^+(i)$ saem de i. Denotamos por $\delta^-(i)$ o conjunto $\{(j, i) \in A : j \in V\}$ e dizemos que as arestas desse conjunto entram em i. Para um vértice i de D, definimos como n_i o grau de i em G.

Para uma dada arborescência geradora T de D, enraizada em r, um vértice $i \in V\backslash\{r\}$ é uma folha se não tem arestas que saem dele. O vértice r é uma folha de T se exatamente uma aresta de T é incidente a r. O problema da arborescência geradora com muitas folhas (PARGMF) consiste em dados D e r encontrar uma arborescência geradora T enraizada em r com número máximo de folhas. É fácil ver que podemos encontrar uma solução ótima do PAGMF resolvendo uma instância do PARGMF. Com essas considerações, associamos as variáveis de decisão $\{y_a \in \mathbb{R}_+^{|A|} : a \in A\}$ e $\{z_i \in \mathbb{R}_+^{|V|} : i \in V\}$ respectivamente com as arestas e vértices de D. O seguinte programa linear inteiro é uma formulação para grafos orientados:

(P)

$$\max \sum_{i \in V} z_i \tag{5.1}$$

$$\text{sujeito a: } \sum_{a \in \delta^-(i)} y_a = 1,\ i \in V\backslash\{r\}, \tag{5.2}$$

$$\sum_{a \in A(S)} y_a \leq |S| - 1,\ S \subset V,\ |S| \geq 2, \tag{5.3}$$

$$\sum_{a \in \delta^+(i)} y_a + (n_i - 1)z_i \leq (n_i - 1),\ i \in V\backslash\{r\}, \tag{5.4}$$

$$\sum_{a \in \delta^+(r)} y_a + (n_r - 1)z_r \leq n_r, \tag{5.5}$$

$$y_a \in \{0, 1\},\ a \in A, \tag{5.6}$$

$$z_i \in \{0, 1\},\ i \in V. \tag{5.7}$$

As equações (5.2) fazem com que todo vértice diferente de r seja alcançável e as inequações (5.3) eliminam subciclos. Lucena et al. em [20] ressaltam que as inequações (5.2), (5.3) e (5.6) descrevem o politopo das arborescências geradoras. Adicionalmente, as inequações (5.4), (5.5) e (5.7) caracterizam folhas de arborescência geradora, onde $z_i = 1,\ i \in V$ significa que o vértice i é uma folha e caso contrário i não é folha. Podemos ver na inequação (5.4) que para um vértice $i \in V\backslash\{r\}$, se $z_i = 1$ então $\sum_{a \in \delta^+(i)} y_a \leq 0$, ou seja, se i é folha então ele não tem aresta de saída. Pela inequação (5.5) se $z_r = 1$ então $\sum_{a \in \delta^+(r)} y_a \leq 1$, ou seja, se r é folha então seu grau de saída é igual a um.

5.1.2 Fortalecendo o modelo

Com exceção do vértice r, as folhas de uma arborescência geradora D, enraizada em r, não podem ter arestas de saída. Logo, a inequação

$$y_a + z_i \leq 1,\ a = (i, j) \in A,\ i \in V\backslash\{r\}, \tag{5.8}$$

é válida para (P). O modelo (P) ainda pode ser reforçado com as seguintes inequações válidas

$$\sum_{a\in F} y_a + \sum_{a=(j,i)\in A:(i,j)\in F} y_a + (|F|-1)z_i \leq |F|,\ i \in V,\ F \subseteq \delta^+(i) \cup \{(r,i)\},\ |F| \geq 2 \quad (5.9)$$

que podem ser vistas como uma generalização de (5.4). Fujie em [10] mostrou que a versão para grafos não orientados das inequações (5.9) define facetas para a envoltória convexa das soluções viáveis de (5.1).

Experimentos computacionais feitos em [20] mostraram que a inclusão das inequações (5.8) e (5.9) ao modelo tendem a fortalecer o limitante dado pela relaxação de (P).

5.1.3 Separação

Nesta seção descrevemos métodos para separação das desigualdades (5.3) e (5.9). Primeiro apresentamos dois métodos para separação das inequações (5.3).

alguma das inequações de (5.3) foi violada para ser incluída no modelo ou provar que nenhuma foi violada. Isso é o mesmo que resolver o seguinte problema: encontrar um conjunto $S \subset V$, $|S| \geq 2$ tal que $\sum_{a\in A(S)} y_a > |S| - 1$ ou provar que $\sum_{a\in A(S)} y_a \leq |S| - 1$ para todo $S \subset V$, $|S| \geq 2$.

Se a solução atual $(\hat{y}, \hat{z})$ é inteira, então basta verificarmos se existe um caminho de r para cada vértice $i \in V\backslash\{r\}$ que passa apenas pelas arestas $a \in A, \hat{y}_a = 1$. Esse caso pode ser resolvido com uma busca em profundidade. Vamos supor que temos uma solução fracionária. Podemos resolver o problema por programação linear inteira. A seguir apresentamos um modelo de programa linear inteiro descrito em [21] para o problema.

Sejam $s \in \{0,1\}^{|V|}$ um vetor de incidência de um conjunto $S \subset V$ e $w \in \{0,1\}^{|A|}$ um vetor de incidência de $A(S)$. Observe que $w \in \{0,1\}^{|A|}$ é um vetor de incidência de $A(S)$ se e somente se $w_{ij} = 1$ quando $s_i = 1$ e $s_j = 1$, e caso contrário $w_{ij} = 0$. Podemos expressar através da seguinte equação não linear:

$$\begin{aligned} &w_a = s_i s_j,\ \text{para todo } a = (i,j) \in A, \\ &s \in \{0,1\}^{|V|}, w \in \{0,1\}^{|A|} \end{aligned} \quad (5.10)$$

que pode ser linealizada

$$\begin{aligned} &w_a \leq s_i, w_a \leq s_j,\ \text{para todo } a = (i,j) \in A, \\ &w_a \geq s_i + s_j - 1,\ \text{para todo } a = (i,j) \in A, \\ &s \in \{0,1\}^{|V|}, w \in \{0,1\}^{|A|}. \end{aligned} \quad (5.11)$$

Denote por $SUB(V,A)$ o conjunto de soluções viáveis de (5.11). Então $(\bar{s}, \bar{w})$ identifica uma restrição de eliminação de subciclo associada ao conjunto $\bar{S}$ e é violada por $\bar{y}$ se e somente se $\bar{s}$ é o vetor incidente de $\bar{S}$, $(\bar{s}, \bar{w}) \in SUB(V,A)$ e $\sum_{a\in A(\bar{S})} \bar{y}_a > |\bar{S}| - 1$.

Observando que $\sum_{a\in A(\bar{S})} \bar{y}_a = \sum_{a\in A(S)} \bar{y}_a\bar{w}_a$ e $|\bar{S}| = \sum_{i\in V} \bar{s}_i$, podemos reescrever a última condição como

$$\sum_{a\in A(S)} \bar{y}_a\bar{w}_a > \sum_{i\in V} \bar{s}_i - 1. \tag{5.12}$$

Finalmente podemos escrever como separar as restrições de eliminação de subciclo. Definimos o seguinte problema de programação linear inteira:

$$\begin{array}{l} \max \ \sum_{a\in A(S)} \bar{y}_a\bar{w}_a - \sum_{i\in V} \bar{s}_i \\ \text{s. a: } (s, w) \in SUB(V, A). \end{array} \tag{5.13}$$

Seja (s^*, w^*) uma solução ótima para (5.13), e v^* o valor ótimo. Podemos ver que se $v^* > -1$, então (s^*, w^*) identifica uma restrição de eliminação de subciclo violada por $\bar{y}$. Se $v^* \leq -1$, então não existe restrição violada.

O teorema seguinte garante que podemos resolver a relaxação linear de (5.13) que é obtida substituindo as restrições de integralidade pelas seguintes restrições lineares:

$$\begin{array}{l} 0 \leq w_a \leq 1, \text{ para todo } a \in A, \\ 0 \leq s_i \leq 1, \text{ para todo } i \in V. \end{array} \tag{5.14}$$

Teorema 5.1. *A relaxação linear de (5.13) sempre tem uma solução inteira ótima que resolve (5.13).*

A prova desse teorema pode ser encontrada em [26].

Outra maneira de separar (5.3) é através de corte mínimo. Considere a restrição $\sum_{a\in A} y_a = |V| - 1$ que é válida para (5.1) e, sejam $S' \subset V, |S'| \geq 1$ e $\bar{S}' = V\backslash S'$. Então:

$$\begin{array}{lcl} \sum_{a\in A} y_a & = & |V| - 1, \\ \sum_{a\in A(S')} y_a + \sum_{a\in A(\bar{S}')} y_a + \sum_{a\in A\backslash(A(S')\bigcup A(\bar{S}'))} y_a & = & |V| - 1, \\ |S'| - 1 + |\bar{S}'| - 1 + \sum_{a\in A\backslash(A(S')\bigcup A(\bar{S}'))} y_a & \geq & |V| - 1, \\ \sum_{a\in A\backslash(A(S')\bigcup A(\bar{S}'))} y_a & \geq & 1. \end{array} \tag{5.15}$$

Logo as restrições (5.3) podem ser substituídas por $\sum_{a\in A\backslash(A(S')\bigcup A(\bar{S}'))} y_a \geq 1$. Para encontrar uma inequação violada basta encontrar o corte mínimo global no grafo orientado e verificar se o valor do corte é menor do que 1.

Apresentamos a seguir um método para separação das inequações (5.9) que é uma adaptação para grafos orientados da sugestão de separação descrita em [20]. Sejam $(\bar{y}, \bar{z})$ um ponto em R e $H_i = \delta^+(i) \cup \{(r, i)\}$, para todo $i \in V$. Para cada $i \in V$, ordene os elementos em $\{\bar{x}_a = \bar{y}_a + \bar{y}_b : a \in H_i, b = (i, j) \in A : (j, i) \in H_i\}$ por ordem decrescente como $\{\bar{x}_{a_1}, \ldots, \bar{x}_{n_i}\}$. Então, para $(\bar{y}, \bar{z})$, $i \in V$ e para todo $l \in \{2, \ldots, n_i\}$, calcule $(\sum_{j=1,l} \bar{x}_{a_j} + (l - 1)\bar{z}_i)$. Ao fazer isso, identificamos para i um conjunto F de

cardinalidade l que atinge o maior valor possível do lado esquerdo de (5.9). Se esse valor é maior do que $|F|$, uma inequação com a maior violação possível para i e l foi identificada. Caso contrário, não existe inequação violada em (5.9) para a combinação i e l. Portanto, para qualquer $i \in V$, a separação das inequações (5.9) podem ser realizadas em tempo $O(n_i \; log \; n_i)$.

5.2 Experimentos Computacionais

Um dos objetivos desse trabalho foi verificar o comportamento na prática de algoritmos para o PAGMF. Implementamos os algoritmos apresentados neste trabalho e executamos testes para algumas instâncias de entrada. A seguir discutimos questões de implementações dos algoritmos, apresentamos como as instâncias de teste foram adquiridas e os resultados obtidos.

Os algoritmos foram implementados na linguagem C++ e usamos o resolvedor comercial Cplex na versão 12.4 para resolver o modelo de programação linear do algoritmo exato. Os códigos usados estão disponíveis em `http://ic.unicamp.br/~ra115137/diss`. Os testes foram executados em uma máquina com processador Intel Core i5 com 2,67 GHz, 4 GB de memória RAM e sistema operacional Ubuntu 10.04 versão 32 bits.

5.2.1 Instâncias Testadas

As instâncias de teste foram separadas em dois grupos. Um para o caso geral e outro para grafos cúbicos. As instâncias para o caso geral são as mesmas usadas em [20] e elas foram geradas pela estratégia descrita a seguir. No primeiro passo, é construído um caminho Hamiltoniano aleatório para o grafo G de entrada e assim a conexidade do grafo é garantida. Isso é feito determinando uma permutação aleatória dos vértices e dessa forma são obtidas as primeiras $(|V(G)| - 1)$ arestas. No segundo passo, são adicionadas arestas aleatórias até que uma densidade predefinida para esse grafo seja atingida. Foram geradas um total de 41 instâncias que têm de 30 a 200 vértices e densidades que variam de 5 a 70%.

Apresentamos a seguir como os grafos cúbicos utilizados nos casos de teste foram gerados. Seja n o número de vértices do grafo cúbico. Primeiro é construído uma árvore geradora T com n vértices tal que cada vértice tem grau máximo 3. A árvore é construída iterativamente, onde em cada iteração uma aresta xy é adicionada à árvore, tal que, x é um vértice aleatório que pertence a árvore e tem grau menor do que 3 e y é um vértice aleatório que ainda não está na árvore. O primeiro vértice adicionado à árvore é escolhido de forma aleatória. Em seguida é gerado um grafo cúbico G' a partir de T através da adição de arestas. Por fim são retirados arestas múltiplas e laços de G' de forma que o

grafo resultante G permaneça cúbico. Geramos um total de 42 instâncias que têm de 30 a 200 vértices.

As instâncias geradas para os grafos cúbicos tinham no máximo 4 triângulos e diamantes. Resolvemos gerar instâncias para os grafos cúbicos com uma quantidade mínima arbitrária de triângulos e diamantes para verificarmos se isso afetaria os resultados. Definimos que a quantidade mínima de vértices do grafo que pertenceria a triângulos ou diamantes seria de aproximadamente 50% e as instâncias foram geradas da seguinte maneira. Primeiro geramos um grafo cúbico com n vértices como definido no parágrafo anterior. Em seguida adicionamos triângulos e diamantes ao grafo. Os triângulos são adicionados escolhendo um vértice aleatório e substituindo-o por um triângulo. Já os diamantes são adicionados entre um par de vértices uv, tal que, uv é aresta do grafo. Veja um exemplo na Figura 5.1, onde temos um exemplo de inserção de triângulo em a) e de diamante em b).

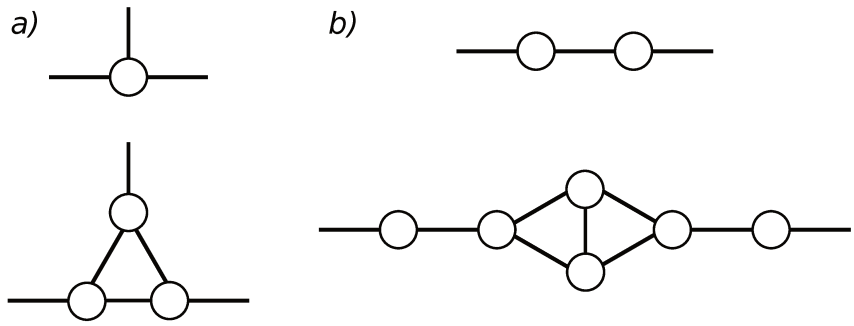


Figura 5.1: Exemplo de inserção de triângulo e diamante.

5.2.2 Detalhes de implementação

Nesta seção discutimos alguns detalhes de implementação dos algoritmos testados. Começamos pelo algoritmo exato, onde usamos dois fatos para fazer um pré-processamento e tentar reduzir o espaço de busca. O primeiro fato é que um vértice de G com grau 1 será uma folha em qualquer árvore geradora de G. Por outro lado, um vértice i de G que desconecta o grafo G se for removido será um vértice interno em qualquer ávore geradora de G. Considerando esses fatos nós fixamos os valores de algumas variáveis quando possível.

Como apresentado na seção 5.1.1 para construção do grafo orientado D é necessário a escolha de um vértice raiz r. Na nossa implementação r é um vértice com maior grau em G.

Usamos um algoritmo guloso para encontrar uma solução inicial que é ligeiramente diferente do proposto por Lucena *et. al.* em [20]. Construímos uma árvore geradora T

de G, que começa com um vértice de maior grau em G e seus vizinhos. Iterativamente escolhemos uma folha f em T, tal que, f tenha o maior número de vizinhos que não pertencem a T. Adicionamos esses vizinhos a T e o processo é repetido até que T seja uma árvore geradora de G.

Para separação das desigualdades (5.3) usamos o primeiro método apresentado na seção 5.1.3 pois para esses casos de teste os resultados foram ligeiramente melhores do que quando usamos o corte mínimo.

Utilizamos dois critérios de parada para a execução do algoritmo. Determinamos um limite de tempo de 1200 segundos para cada execução. Definimos como $1 - epsilon$ o valor absoluto de tolerância da diferença entre a melhor solução inteira encontrada e o valor objetivo do melhor nó restante, onde *epsilon* é igual a 10^{-6}. Podemos fazer isso já que nossas variáveis e coeficientes são inteiros. Dessa forma, se essa diferença é menor do que $1 - epsilon$, então a melhor solução inteira encontrada é uma solução ótima.

Para o algoritmo 5-aproximado, a árvore geradora inicial foi construída através de uma estratégia de busca em profundidade iniciada no vértice de menor rótulo. Os algoritmos 3 e 2-aproximados têm implementação similar. A diferença é que na busca por folhas que podem ser expandidas o algoritmo 2-aproximado só retorna uma folha de prioridade 1 se não há prioridade 2 e o algoritmo 3-aproximado não observa prioridades.

Em relação ao algoritmo 7/4-aproximado temos as seguintes considerações. A raiz inicial r_0 escolhida foi o vértice com rótulo 1. A aplicação das regras para expansão foi feita de maneira recursiva pois foi a forma que encontramos para poupar computação desnecessária.

Para o algoritmo 3/2-aproximado na fase 1 identificamos e armazenamos todas as T-pontes para depois verificarmos cada uma delas. Na fase 2 identificamos e armazenamos todos os triângulos do tipo 3 e diamantes do tipo 2 para verificarmos em seguida cada um deles. O processo para remoção dos vértices da fase 3 foi o seguinte. Primeiro removemos um vértice de S (S') e em seguida verificamos se o conjunto resultante ainda é um 2-CD-set (CD-set), se for, então consideramos que esse vértice foi removido, caso contrário adicionamos o vértice novamente ao conjunto. Para construção do 2-CD-set minimal também consideramos tentar remover dois vértices ao mesmo tempo de maneira similar. Não construímos a árvore geradora porque nosso maior interesse é o número de folhas da solução e sabemos que é $|V| - |S'|$.

5.2.3 Resultados

Um resumo dos resultados obtidos para o caso geral se encontra na Tabela 5.1. A primeira coluna identifica os algoritmos. Os valores nas colunas “média” e “máximo” são razões do melhor limitante encontrado pelo valor da solução do algoritmo, onde na coluna “média”

esse é o valor da média aritmética, e na coluna "máximo" esse foi o maior valor encontrado. Os valores na coluna "ótimo" indicam a quantidade de vezes em que o algoritmo encontrou uma solução ótima.

Tabela 5.1: Comparação caso geral com melhor limitante encontrado.

Algoritmo	média	máximo	ótimo
5-aprox.	1,09	1,19	1
3-aprox.	1,10	1,30	1
2-aprox.	1,08	1,23	3
guloso	1,02	1,08	16

Para esses casos de teste o algoritmo guloso, em geral, obteve os melhores resultados. Como podemos ver na Tabela 5.1 o algoritmo guloso conseguiu encontrar uma solução ótima em 16 testes do total de 41.

Os resultados obtidos estão na Tabela 5.2. As colunas da Tabela 5.2 são divididas em blocos. O primeiro bloco contém as duas primeiras colunas e identificam as instâncias. A primeira coluna indica a quantidade de vértices no grafo G para cada caso de teste e a segunda a densidade do grafo. O segundo bloco tem três colunas consecutivas referentes ao algoritmo exato. A primeira delas contém o valor do melhor limitante dual e a segunda o valor da melhor solução inteira encontrada. A terceira coluna contém o tempo gasto em segundos pelo algoritmo, onde "-" representa o tempo máximo de execução estipulado. O último bloco tem nas três primeiras colunas os valores das soluções encontradas pelos algoritmos aproximados para o caso geral. A primeira é referente ao algoritmo 5-aproximado, a segunda ao 3-aproximado e a terceira ao 2-aproximado. A última coluna tem os valores das soluções encontradas pelo algoritmo guloso que gera uma solução inicial para o problema.

Podemos observar que o algoritmo exato encontra uma solução ótima para todas as instâncias com 30, 50 e 70 vértices, e que a partir daí tem dificuldade para encontrar uma solução ótima, ao ponto que para as instâncias de 150 e 200 vértices não encontrou nenhuma solução ótima no tempo máximo determinado. Além disso, observando as colunas dos melhores limitantes primais e duais encontrados, podemos afirmar que para essas instâncias, quanto menor a densidade do grafos mais difícil é encontrar uma solução ótima. Outra observação é que nenhuma das soluções encontradas pelos algoritmos aproximados tiveram mais folhas do que as soluções do algoritmo guloso.

Os resultados obtidos da execução do algoritmo exato, apesar do pouco número de soluções ótimas encontradas, estão coerentes com os resultados encontrados por [20]. Acreditamos que os resultados não foram melhores devido ao limite de tempo de execução.

Um resumo dos resultados obtidos para os grafos cúbicos estão na Tabela 5.3. A primeira coluna identifica os algoritmos. Os valores nas colunas "média", "média ted",

"máximo" e "máximo ted" são razões do melhor limitante encontrado pelo valor da solução do algoritmo, onde nas colunas "média" e "média ted" esses são os valores da média aritmética, e nas colunas "máximo" e "máximo ted" encontram-se os maiores valores encontrados. As colunas "média" e "máximo" são referentes às instâncias geradas de grafos cúbicos, e as colunas "média ted" e "máximo ted" são referentes às instâncias de grafos cúbicos com número mínimo de triângulos e diamantes.

Podemos observar na coluna média da Tabela 5.3 que o algoritmo 2-aproximado obteve os melhores resultados em média, seguido do algoritmo 7/4-aproximado. O melhor algoritmo para as instâncias com número mínimo definido de triângulos e diamantes foi o algoritmo 3/2-aproximado, ou seja, quando aumentamos o número de triângulos e diamantes um algoritmo específico teve desempenho melhor do que um algoritmo para o caso geral.

Os resultados obtidos para os grafos cúbicos estão na Tabela 5.4. As colunas da Tabela 5.4 são divididas em blocos. O primeiro bloco contém as duas primeiras colunas e identificam as instâncias. A primeira coluna indica a quantidade de vértices no grafo G para cada caso de teste e a segunda o número da instância. O segundo bloco tem três colunas consecutivas referentes ao algoritmo exato. A primeira delas contém o valor do melhor limitante primal e a segunda o valor da melhor solução inteira encontrada. A terceira coluna contém o tempo gasto em segundos pelo algoritmo, onde - representa o tempo máximo de execução estipulado. O último bloco tem nas três primeiras colunas os valores das soluções encontradas pelos algoritmos aproximados para o caso geral. A primeira é referente ao algoritmo 5-aproximado, a segunda ao 3-aproximado e a terceira ao 2-aproximado. A coluna seguinte tem os valores das soluções encontradas pelo algoritmo guloso que gera uma solução inicial para o problema. As duas últimas colunas têm os valores das soluções encontradas pelos algoritmos 7/4-aproximado e 3/2-aproximado.

Na Tabela 5.4 podemos ver que o algoritmo 2-aproximado obtém os melhores resultados em média. Além disso, podemos notar que o algoritmo exato encontrou soluções ótimas apenas para as instâncias com até 50 vértices.

Tabela 5.2: Resultados para o caso geral.

$\|V\|$	d	LD	LP	Tempo (s)	5-aprox.	3-aprox.	2-aprox.	guloso
30	10	15,92	15	0,03	15	12	15	15
	20	23,88	23	0,22	21	20	20	23
	30	26,47	26	0,40	24	23	23	25
	50	27,90	27	0,42	25	25	25	27
	70	28,79	28	0,02	26	28	28	28
50	5	19,80	19	0,08	17	18	19	19
	10	38,44	38	11,77	35	33	33	36
	20	43,98	43	1,96	41	39	39	43
	30	45,99	45	2,23	38	41	42	45
	50	47,94	47	1,97	44	45	45	47
	70	48,69	48	0,05	45	46	46	48
70	5	43,99	43	69,62	40	36	37	40
	10	57,99	57	3,08	50	47	47	56
	20	63,99	63	16,48	53	55	55	62
	30	65,99	65	11,45	60	60	60	65
	50	67,99	67	10,86	62	64	64	67
	70	68,51	68	274,30	66	66	66	67
100	5	77,49	75	-	66	62	65	72
	10	87,99	87	12,64	78	78	78	87
	20	92,99	92	144,68	85	86	86	92
	30	95,51	94	-	87	89	89	93
	50	96,99	96	477,83	93	94	94	96
	70	97,98	97	177,26	94	95	95	97
120	5	96,54	92	-	84	74	78	91
	10	107,91	106	-	96	96	97	105
	20	113,22	112	-	100	102	103	111
	30	115,04	114	-	107	109	109	114
	50	117,12	116	-	113	113	113	116
	70	117,99	117	413,92	115	115	115	117
150	5	127,37	121	-	107	104	105	119
	10	137,87	134	-	123	121	123	134
	20	143,61	141	-	131	133	134	140
	30	145,82	143	-	137	138	138	143
	50	147,17	146	-	141	142	142	146
	70	148,45	147	-	143	145	145	147
200	5	177,29	168	-	150	150	150	168
	10	188,24	182	-	167	168	169	182
	20	194,29	190	-	181	182	182	190
	30	196,12	193	-	187	187	187	193
	50	198,01	195	-	192	191	191	195
	70	198,55	197	-	193	196	196	197

Tabela 5.3: Comparação grafos cúbicos com melhor limitante encontrado.

Algoritmo	média	média ted	máximo	máximo ted
5-aprox.	1,15	1,22	1,30	1,44
3-aprox.	1,16	1,15	1,24	1,25
2-aprox.	1,07	1,12	1,13	1,21
guloso	1,10	1,14	1,20	1,27
7/4-aprox.	1,08	1,13	1,14	1,21
3/2-aprox.	1,11	1,11	1,23	1,18

Tabela 5.4: Resultados para o caso grafos cúbicos.

$\|V\|$	i	LD	LP	Tempo (s)	5-ap.	3-ap.	2-ap.	guloso	7/4-ap.	3/2-ap.
30	1	16	16	0,69	14	14	15	15	14	15
	2	16	16	0,88	13	13	15	14	15	15
	3	16	16	0,05	13	15	15	15	14	14
	4	16	16	1,18	15	14	15	15	16	15
	5	16	16	0,01	13	13	16	16	16	13
	6	16	16	0,31	13	15	15	15	14	14
50	1	26	26	1177,54	23	23	24	23	24	23
	2	26	26	855,79	22	24	24	22	24	23
	3	26	26	337,45	23	24	24	23	25	23
	4	26	26	426,16	24	24	24	24	24	23
	5	26	26	104,81	24	23	24	23	24	24
	6	26	26	123,94	20	23	23	23	24	24
70	1	36	34	-	33	30	34	34	33	33
	2	36	33	-	34	29	33	31	33	32
	3	36	32	-	30	30	35	32	34	33
	4	36	35	-	30	32	35	35	34	31
	5	36	33	-	30	31	34	33	35	31
	6	36	31	-	31	31	34	30	32	33
100	1	51	48	-	44	42	49	48	47	46
	2	51	47	-	42	44	49	47	47	48
	3	51	44	-	47	44	46	44	47	45
	4	51	46	-	47	41	48	46	47	46
	5	51	45	-	44	42	46	45	48	47
	6	51	46	-	46	44	48	46	47	47
120	1	61	56	-	53	55	58	56	57	55
	2	61	53	-	55	53	56	53	57	54
	3	61	56	-	53	51	59	56	56	56
	4	61	55	-	55	52	57	55	56	56
	5	61	57	-	53	52	58	57	57	54
	6	61	58	-	56	52	58	58	57	56
150	1	76	70	-	69	67	71	70	71	70
	2	76	66	-	68	69	70	66	72	68
	3	76	71	-	66	65	71	71	71	69
	4	76	70	-	67	65	71	70	70	70
	5	76	70	-	66	65	71	70	71	73
	6	76	70	-	67	66	72	70	71	70
200	1	101	88	-	91	85	94	88	94	93
	2	101	89	-	86	85	93	89	92	93
	3	101	88	-	90	86	92	88	97	88
	4	101	94	-	91	86	94	94	96	93
	5	101	90	-	88	88	94	90	96	90
	6	101	92	-	85	87	95	92	94	93

Capítulo 6

Conclusões

O principal objetivo deste trabalho foi verificar o comportamento na prática dos algoritmos aproximados para o PAGMF e para o caso particular onde temos apenas grafos cúbicos. Para isso fizemos um estudo teórico dos algoritmos encontrados na literatura, implementamos alguns desses algoritmos, testamos instâncias utilizadas na literatura e outras geradas neste trabalho. Implementamos um algoritmo exato com objetivo de termos um limitante superior para ser usado na comparação dos algoritmos aproximados.

Executamos testes onde os grafos de entrada tinham de 30 a 200 vértices e conseguimos encontrar soluções ótimas de instâncias com até 120 vértices para o PAGMF e 50 vértices para os grafos cúbicos. Para os casos de teste, os grafos com menor densidade de arestas se mostraram mais difíceis de serem resolvidos de forma exata. Os resultados mostraram que o algoritmo guloso foi o melhor para o PAGMF e o algoritmo 2-aproximado foi o melhor para os grafos cúbicos. Sendo que o algoritmo guloso foi muito melhor do que os outros no PAGMF. Nos grafos cúbicos o 2-aproximado foi o melhor, mas a diferença em média foi pequena em relação ao segundo melhor.

Como trabalho futuro, relacionamos algumas sugestões. Seria interessante verificar se há uma razão de aproximação do algoritmo guloso apresentado na seção 5.2.2. Outra sugestão é usar o algoritmo 5-aproximado para fornecer uma solução inicial para o algoritmo exato, mas fornecendo como árvore inicial para o 5-aproximado uma solução do algoritmo guloso. Uma outra possível tarefa, seria explorar a escolha dos vértices nos algoritmos aproximados levando em conta o grau de cada um. A intuição é que os resultados na prática poderiam melhorar devido ao bom desempenho do algoritmo guloso.

Referências Bibliográficas

[1] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Macmillan, 1976.

[2] P. Bonsma. Max-leaves spanning tree is APX-hard for cubic graphs. *Journal of Discrete Algorithms*, 12:14–23, 2012.

[3] P. Bonsma and F. Zickfeld. A 3/2-approximation algorithm for finding spanning trees with many leaves in cubic graphs. *Lecture Notes in Computer Science*, 5344:66–77, 2008.

[4] M. H. Carvalho, M. R. Cerioli, R. Dahab, P. Feofiloff, C. G. Fernandes, C. E. Ferreira, F. K. Miyazawa, J. C. de Pina, J. Soares, and Y. Wakabayashi. *Uma Introdução Sucinta a Algoritmos de Aproximação*. Publicações Matemáticas do IMPA, 2001.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[6] J. R. Correa, C. G. Fernandes, M. Matamala, and Y. Wakabayashi. A 5/3-approximation for finding spanning trees with many leaves in cubic graphs. *Lecture Notes in Computer Science*, 4927:184–192, 2008.

[7] J. Daligault and S. Thomassé. On finding directed trees with many leaves. *Lecture Notes in Computer Science*, 5917:86–97, 2009.

[8] M. Drescher and A. Vetta. An approximation algorithm for the maximum leaf spanning arborescence problem. *ACM Transactions on Algorithms*, 6(3), Junho 2010.

[9] T. Fujie. An exact algorithm for the maximum leaf spanning tree problem. *Computers & Operations Research*, 30:1931–1944, 2003.

[10] T. Fujie. The maximum-leaf spanning tree problem: formulation and facets. *Networks*, 43(4):212–223, 2004.

[11] E. G. Fusco and A. Monti. Spanning trees with many leaves in regular bipartite graphs. *Lecture Notes in Computer Science*, 4835:904–914, 2007.

[12] G. Galbiati, F. Maffioli, and A. Morzenti. A short note on the approximability of the maximum leaves spanning tree problem. *Information Processing Letters*, 52:45–49, 1994.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman, San Francisco, CA, USA, 1979.

[14] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[15] P. Lemke. The maximum-leaf spanning tree problem in cubic graphs is NP-complete. *IMA Preprint Series*, (428), July 1988.

[16] (Ben) P. C. Li and M. Toulouse. Variations of the maximum leaf spanning tree problem for bipartite graphs. *Information Processing Letters*, 97(4):129–132, 2006.

[17] K. Lorys and G. Zwozniak. Approximation algorithm for the maximum leaf spanning tree problem for cubic graphs. *Lecture Notes in Computer Science*, 2461:686–697, 2002.

[18] H. Lu and R. Ravi. The power of local optimization: Approximation algorithms for maximum-leaf spanning tree. In *Thirtieth Annual Allerton Conference on Communication, Control and Computing*, pages 533–542, 1996.

[19] H. Lu and R. Ravi. Approximating maximum leaf spanning trees in almost linear time. *Journal of Algorithms*, 29:132–141, 1998.

[20] A. Lucena, N. Maculan, and L. Simonetti. Reformulations and solution algorithms for the maximum leaf spanning tree problem. *Computational Management Science*, 7(3):289–311, 2010.

[21] C. Mannino and G. Dahl. Notes on combinatorial optimization, 2012. Disponível em: http://heim.ifi.uio.no/∼geird/comb_notes.pdf. Acessado em 15/01/2013.

[22] M. S. Rahman and M. Kaykobad. Complexities of some interesting problems on spanning trees. *Information Processing Letters*, 94(2):93–97, 2005.

[23] N. Schwartges, J. Spoerhase, and A. Wolff. Approximation algorithms for the maximum leaf spanning tree problem on acyclic digraphs. *Lecture Notes in Computer Science*, 7164:77–88, 2012.

[24] R. Solis-Oba. 2-approximation algorithm for finding a spanning tree with maximum number of leaves. *Lecture Notes in Computer Science*, 1461:441–452, 1998.

[25] J. A. Storer. Constructing full spanning trees for cubic graphs. *Information Processing Letters*, pages 8–11, 1981.

[26] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, 1998.