Decomposição em Árvore de Grafos com Largura Limitada — Uma Pesquisa Algorítmica

Luis Eduardo Ximenes Carvalho

Departamento de Ciência da Computação - DC/UFC
Universidade Federal do Ceará
Fortaleza, CE, Brasil

Agosto, 2002

Versão parcial de Dissertação a ser apresentada ao
Mestrado em Ciência da Computação,
UFC, como requisito parcial para a
obtenção do título de Mestre em
Ciência da Computação.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Durante cerca de uma década, Neil Robertson e Paul Seymour provaram um dos mais importantes resultados em Matemática Discreta: "*em todo conjunto infinito de grafos, existem dois tais que um é menor do outro*". Tal teorema é conhecido como *Teorema de Menores de Grafos*, e sua prova toma cerca de 500 páginas, sendo coberta por sua célebre série de artigos no *Journal of Combinatorial Theory, Series B*, entre 1986 e 1997.

Como resultado parcial do esforço da prova do teorema, desenvolveram-se diversos conceitos e técnicas de grande potencial e interesse. Um desses conceitos é o de decomposição em árvore. Uma decomposição em árvore para um grafo procura estabelecer uma relação estrutural entre as partes deste de modo que estas estejam minimalmente conectadas, ou seja, de modo que a estrutura do grafo assemelhe-se a de uma árvore. Este tipo de decomposição constitui, desta forma, uma boa ferramenta para o desenvolvimento de algoritmos eficientes para muitos problemas de otimização combinatória que sejam modelados em grafos, dado que tais grafos obedeçam uma estrutura bastante semelhante a uma árvore. O critério para tal similaridade é conhecido como largura em árvore: quanto menor a largura em árvore de um grafo, mais semelhante este é a uma árvore; ao contrário, quanto maior a largura em árvore, mais conexo o grafo será.

Muitos pesquisadores voltaram então seus esforços para a caracterização de tipos de grafos que sejam suficientemente semelhantes a uma árvore para que essa propriedade possa ser usada eficientemente, principalmente no projeto de algoritmos específicos. Tais grafos compõem a classe dos grafos que possuem largura em árvore limitada. No entanto, embora este seja atualmente um assunto bastante estudado e pesquisado em todo o mundo, é bastante incipiente no Brasil o desenvolvimento de iniciativas e contribuições nesse sentido.

Este texto compreende uma contribuição para este campo de estudo em Teoria dos Grafos, de modo principalmente a estimular o surgimento de novos trabalhos e estudos de profundidade, compatíveis com sua complexidade. Este texto pretende assumir um caráter introdutório, sendo redigido em linguagem acessível, e, na medida do possível,

auto-contido e de fácil compreensão. Além disso, é também intenção do texto ilustrar o estado da arte em algoritmos de decomposição em árvore. Embora este tenha sido o principal objetivo do trabalho de pesquisa, o principal produto é o texto, sendo este bastante trabalhado.

Uma qualidade do texto que foi valorizada é o equilíbrio entre o formalismo matemático necessário e uma redação mais leve, introdutória. Tal aspecto teve como objetivo uma assimilação mais suave da teoria, que pode, numa primeira impressão, parecer intratável e hermética. No entanto, vale ressaltar que a familiaridade crescente com o assunto requer, invariavelmente, um bom envolvimento e compreensão dos conceitos e resultados, e principalmente das provas. De fato, buscou-se uma abordagem compreensiva e completa, e praticamente todos os principais resultados da teoria são demonstrados — com algumas exceções nos casos em que a prova é muito longa ou técnica. O melhor aprendizado passa, sem dúvida, pela leitura minuciosa e pelo estudo judicioso dos lemas, teoremas e definições e suas provas. Nesse sentido, procurou-se também fornecer provas mais simples e detalhadas, principalmente dos resultados mais importantes.

O texto procura manter a simplicidade inclusive em sua organização, contendo somente três capítulos principais. O Capítulo 2 introduz os conceitos básicos necessários em Teoria dos Grafos. No Capítulo 3, o tema principal do texto é discutido: são apresentados conceitos de decomposição em árvore e largura em árvore, com algumas propriedades; em seguida casos particulares de classes de grafos com largura em árvore limitada e pequena são tratados; e, ao final do capítulo, é feita uma discussão de novos conceitos de conectividade para que se possa estabelecer uma obstrução estrutural para grafos com largura em árvore limitada.

O Capítulo 4 é dedicado aos algoritmos para obtenção de decomposições em árvore. Nele são abordados dois paradigmas principais: algoritmos que usam métodos recursivos e construtivos, e algoritmos baseados em redução. Para tanto, são discutidos três principais algoritmos, compondo os mais importantes e atuais resultados tanto em termos de técnicas como de complexidade computacional. Finalmente, no Capítulo 5, são articuladas algumas conclusões e recomendações para trabalhos futuros.

Capítulo 2

Conceitos e Preliminares

2.1 Introdução

O objetivo deste capítulo é introduzir os principais conceitos a serem usados ao longo do texto, assim como apresentar a notação, tornando-a mais familiar para o leitor. É assumido que este tenha alguma familiaridade com Teoria dos Grafos e Algoritmos. Os conceitos foram retirados de várias fontes, principalmente de (Bondy e Murty 1976), (Golumbic 1980), (Diestel 2000), mas compartilham a notação adotada por este último.

2.2 Grafos

Definição 1 (Grafo). Um *grafo* G é um par (V, E), onde V denota o *conjunto de vértices*, e E o *conjunto de arestas* de G. Cada aresta é um par não-ordenado de vértices $u, v \in V$, denotada por (u, v).

Os conjuntos de vértices e de arestas de um grafo G são denotados, respectivamente, por $V(G)$ e $E(G)$. Um grafo *finito* é um grafo tal que seus conjuntos de vértices e arestas são finitos; a cardinalidade de $V(G)$ é normalmente denotada por n e a de $E(G)$ por m. Um *laço* é uma aresta (u, v) onde $u = v$, e arestas que compartilham os mesmos vértices, ou seja $e_1, \ldots, e_k = (u, v)$, são chamadas *múltiplas* entre u e v ou *paralelas*. Um grafo *simples* G é um grafo sem laços e arestas múltiplas, ou seja, para qualquer aresta (u, v) de G, $u, v \in V(G)$, $u \neq v$. Ao longo do texto, um *grafo* refere-se a um grafo simples finito, a menos que seja expresso o contrário. Um *grafo trivial* é um que contém apenas um vértice, $(\{v\}, \emptyset)$, enquanto que um *grafo vazio* contém ambos os conjuntos de vértices e arestas vazios.

Seja $G = (V, E)$ um grafo. Para qualquer aresta $e = (u, v) \in E$, u e v são chamados *extremidades* de e, e e é chamada a aresta entre u e v, ou conectando u e v. Dois vértices $u, v \in V$ são *adjacentes* se existe uma aresta $(u, v) \in E$. Se dois vértices u e v são

adjacentes, também diz-se que u é *vizinho* de v e vice versa. Um vértice $v \in V$ e uma aresta $e \in E$ são chamados *incidentes* se $e = (u, v)$ para algum $u \in V$. Duas arestas são ditas *adjacentes* se compartilham uma mesma extremidade. A *vizinhança* de um vértice v, denotada por $N_G(v)$, corresponde ao conjunto de vértices adjacentes a v: $N_G(v) = \{u \in V : (u, v) \in E\}$. A vizinhança de um conjunto de vértices S, $S \subseteq V$, denotada por $N_G(S)$, corresponde ao conjunto de todos os vértices adjacentes a algum elemento de S e não contidos nele: $N_G(S) = \{v : v \in N_G(u), u \in S, v \notin S\}$. O *grau* de um vértice v em G é o número de arestas que são incidentes a v, sendo denotado por $d(v)$. Os *graus mínimo* e *máximo* de G, denotados por $\delta(G)$ e $\Delta(G)$, correspondem ao mínimo e máximo dos graus dentre todos os vértices de G, respectivamente, $\delta(G) = \min_{v \in V}\{d(v)\}$ e $\Delta(G) = \max_{v \in V}\{d(v)\}$. Para um grafo G com arestas múltiplas, o grafo simples G' obtido pela remoção de arestas múltiplas entre quaisquer vértices u e v, e posterior adição de (u, v) a $E(G')$, é conhecido como *grafo subjacente* a G.

Um grafo G' é um *subgrafo* de um grafo G se $V(G') \subseteq V(G)$ e $E(G') \subseteq E(G)$. Se G' é um subgrafo de G, então G é chamado um *supergrafo* de G'. Se um subgrafo G' de G é tal que $V(G') = V(G)$, então G' é dito um *subgrafo gerador* de G. Um grafo G' é um subgrafo de G *induzido (por vértices)* por W, onde $W \subseteq V(G)$, se $V(G') = W$ e $E(G') = \{(u, v) \in E : u, v \in W\}$. Diz-se também que G' é um *subgrafo induzido* de G. Para qualquer $W \subseteq V(G)$, o subgrafo induzido por W é denotado por $G[W]$. Analogamente, se $F \subseteq E$, diz-se que G' é um *subgrafo induzido em arestas* por F de G se $E(G') = F$ e $V(G') = \{u \in V : (u, v) \in F, v \in V\}$. Da mesma forma, denota-se G' por $G[F]$.

A definição de subgrafo pode ser vista como uma relação entre o grafo original e seu subgrafo. De forma semelhante, algumas operações podem ser definidas para grafos. A *união* de dois grafos $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$, $G_1 \cup G_2$, resulta num grafo $G = (V, E)$ tal que $V = V_1 \cup V_2$ e $E = E_1 \cup E_2$. A *adição* de um vértice v a um grafo G pode ser vista como um caso particular da união entre G e um grafo trivial contendo v, sendo denotada simplesmente por $G + v$; analogamente, denota-se por $G + e$ a adição de uma aresta $e = (u, v)$ a G, sendo igual a $G \cup H$, $H = (\{u, v\}, \{e\})$. Ao contrário da adição, pode-se retirar um conjunto de vértices X de um grafo G, $X \subseteq V$, com base na diferença usual entre o conjunto de vértices de G e X: $G - X = G[V \setminus X]$. Desta forma, pode-se ainda denotar a retirada de um vértice v ou aresta e por $G - v$ e $G - e$, respectivamente. A *interseção* entre dois grafos G_1 e G_2 é definida analogamente à união, resultando no grafo $G = G_1 \cap G_2$ onde $V = V_1 \cap V_2$ e $E = E_1 \cap E_2$. Se G é um grafo com subgrafos induzidos G_1, G_2 e S tais que $G = G_1 \cup G_2$ e $S = G_1 \cap G_2$, então diz-se que G é obtido pela (operação de) *colagem de G_1 e G_2 ao longo de S*.

Uma *trilha* W em um grafo G é uma seqüência alternada $\langle v_1, e_1, v_2, e_2, \ldots, e_{p-1}, v_p\rangle$ de vértices e arestas $(p \geq 1)$, começando e terminando em um vértice, tal que, para cada i, $v_i \in V$, $e_i \in E$, e $e_i = (v_i, v_{i+1})$. A trilha W é também chamada de trilha de v_1 a v_p ou trilha entre v_1 e v_p. Os vértices v_1 e v_p são chamados extremidades de W, e os outros vértices são chamados internos. O *comprimento* de uma trilha é o número de arestas que esta contém. Diz-se ainda que uma trilha W *usa* um vértice v se $v = v_i$ para algum i, $1 \leq i \leq p$, e que W *evita* v se W não usa v. Se G é um grafo simples, então não há necessidade de referenciar as arestas em W, já que estas tornam-se implicitamente deter-

minadas; neste caso, ou quando somente a seqüência de vértices for relevante, pode-se denotar W por $\langle v_1, \ldots, v_p \rangle$, de forma que para cada i, $1 \leq i \leq p$, $(v_i, v_{i+1}) \in E(G)$.

Um *caminho* P em um grafo G é uma trilha na qual todos os vértices são distintos, e logo todas as arestas são também distintas. A *distância* entre dois vértices u e v em G é o comprimento do menor caminho entre u e v em G. Se P é o caminho mínimo entre u e v em G, então diz-se que P *realiza* a distância entre u e v. Seja $P = \langle x_1, x_2, \ldots, x_{r-1}, x_r \rangle$; então denotam-se os subcaminhos de P $\langle x_1, \ldots, x_i \rangle$, $\langle x_i, \ldots, x_r \rangle$ e $\langle x_i, \ldots, x_j \rangle$, como Px_i, x_iP e x_iPx_j, respectivamente. Analogamente, a união de dois caminhos $P_1 = \langle x_1, \ldots, x_i \rangle$ e $P_2 = \langle x_i, \ldots, x_r \rangle$ resultando no caminho $P = \langle x_1, \ldots, x_r \rangle$ pode ser denotada por $P = P_1x_i \cup x_iP_2$, ou $P = P_1x_iP_2$, ou mesmo $P = \langle x_1, P_1, x_i, P_2, x_r \rangle$.

Um *ciclo* C em G é uma trilha na qual todas as arestas são distintas, e todos os vértices são distintos com exceção das extremidades, ou seja, apenas o primeiro e último vértices são iguais. Se um grafo G consiste de apenas um caminho, então G é dito um caminho em n vértices, sendo denotado por P_n; da mesma forma, se G é composto de um único ciclo, G é dito um ciclo em n vértices, denotado por C_n.

Dois vértices são *conectados* em um grafo G se existe um caminho entre eles. Um grafo G é dito *conexo* se todo par de vértices de G é conectado. Uma *componente (conexa)* C de G é um subgrafo conexo maximal de G, ou seja, C é um subgrafo de G que é conexo, e não existe um subgrafo de G que contenha propriamente C e seja também conexo. Um conjunto $S \subseteq V$ é um *separador* de G se existem dois vértices $u, v \in V \setminus S$ tais que u e v são conectados em G e desconectados em $G[V \setminus S]$; diz-se ainda que S é um uv-separador em G, e que S é um *corte* em G. Se nenhum subconjunto próprio de S é um uv-separador, então S é um *separador minimal* para u e v. Uma *articulação* de G é um vértice $v \in V(G)$ tal que $\{v\}$ é um separador de G. Um grafo G é *biconexo* se G é conexo e não contém articulações. Uma *componente biconexa* ou *bloco* de G é um subgrafo biconexo maximal de G. Pode-se ver que os blocos de G particionam o conjunto de arestas E de G, cada bloco é um subgrafo induzido de G, e um vértice $v \in V$ é uma articulação se e somente se v pertence a dois ou mais blocos de G. Uma aresta $e \in E$ é chamada *ponte* de G se existem dois vértices $u, v \in V$ que são conectados em G mas desconectados em $G' = (V, E \setminus \{e\})$.

Uma *partição* de um conjunto S é uma divisão dos elementos de S em classes disjuntas $\{X_i\}_{i \in I}$ tal que $\bigcup_{i \in I} X_i = S$ e $X_i \cap X_j = \emptyset$ para todo $i, j \in I$, $i \neq j$; uma r-partição é uma partição tal que $|I| = r$. Um grafo G é dito r-particionado se admite uma r-partição de seu conjunto de vértices $V(G)$ tal que toda aresta de G tem extremidades em classes diferentes da partição. Se um grafo G é 2-particionado em conjuntos $X, Y \subseteq V(G)$, por exemplo, então toda aresta $e = (u, v)$ de G satisfaz $u = X$ e $v = Y$. Normalmente se denomina G por grafo *bipartite*.

A *potência* de um conjunto S é a família $\mathcal{P}(S)$ de todos os subconjuntos de S: $\mathcal{P}(S) = \{X : X \subseteq S\}$. Um grafo *completo* é um grafo (simples) G onde todos os vértices são adjacentes entre si, ou seja, para todo $X \in \mathcal{P}(V(G))$, existe uma aresta de G entre cada par de vértices de X. O grafo completo com n vértices é denotado por K_n. Uma *clique* de um grafo G é um subconjunto $S \subseteq V(G)$ tal que $G[S]$ é um subgrafo completo de G. O número clique de G, denotado por $\omega(G)$, é o tamanho da maior clique

em G. Por simplicidade, um grafo completo é também conhecido como uma clique. Um grafo *bipartite completo*, por sua vez, é um grafo bipartite onde quaisquer dois vértices de classes diferentes são adjacentes entre si. A partir do conceito de clique, pode-se ainda definir uma *operação clique* entre um grafo G e um subconjunto de vértices X de G, denotada por $G + K(X)$, resultando num supergrafo G' de G onde os vértices em X formam uma clique.

Dois grafos $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$ são ditos *isomorfos* — sendo denotado por $G_1 \approx G_2$ — se existem bijeções $f : V_1 \mapsto V_2$ e $g : E_1 \mapsto E_2$ tais que para cada $v \in V_1$ e $e \in E_1$, v é incidente a e em G_1 se e somente se $f(v)$ é incidente a $g(e)$ em G_2. O par (f, g) é chamado um *isomorfismo* de G_1 a G_2. Se G_1 e G_2 são grafos simples, um isomorfismo entre eles pode ser representado simplesmente por uma função f: G_1 e G_2 são isomorfos se e somente se existe $f : V_1 \mapsto V_2$ tal que $(u, v) \in E_1$ se e somente se $(f(u), f(v)) \in E_2$.

Diz-se que um grafo é *planar*, ou *embutível no plano* se este pode ser desenhado no plano de forma que suas arestas interceptam-se apenas nas extremidades.

2.3 Árvores

Uma *árvore* T é um grafo simples conexo sem ciclos. Uma *floresta* F é um grafo simples sem ciclos, ou seja, um grafo é uma floresta se e somente se cada uma de suas componentes é uma árvore. Vale notar que qualquer aresta de uma árvore T é uma ponte, e logo T é minimalmente conectado. Um vértice de uma árvore é também particularmente chamado de *nó*.

Lema 2.1. *Em uma árvore T, quaisquer dois nós são conectados por um* único *caminho.*

Prova. Seja T uma árvore e $u, v \in V(T)$ nós de T. Suponha por absurdo que existam pelo menos dois caminhos P_1 e P_2 unindo u e v em T. Basta notar agora que a diferença simétrica entre P_1 e P_2, $(P_1 \cup P_2) \setminus (P_1 \cap P_2)$, contém pelo menos um ciclo, um absurdo, já que, por hipótese, T é uma árvore. Logo, a propriedade segue. □

Pode-se ver que o lema anterior decorre naturalmente do fato de que uma árvore não possui ciclos. Desta forma, ao se tomar um caminho maximal P em uma árvore T, pode-se definir como *folhas* os vértices extremidades de P. As folhas de T são os vértices extremidades dos caminhos maximais de T, ou seja, todos os vértices de grau unitário. Os nós que não são folhas são chamados *internos*, já que também são internos a algum caminho maximal em T. Além disso, pelo Lema 2.1, como o caminho P entre dois nós u e v de uma árvore T é bem definido, pode-se denotar uTv como simplificação para uPv.

Uma *árvore enraizada* é uma árvore T com um nó destacado r chamado *raiz* de T. Numa árvore enraizada T, os *descendentes* de um nó $v \in V(T)$ são os nós u tais que o caminho de u até a raiz r usa v. Os *filhos* de v são os descendentes de v que têm distância unitária a v. Pelo Lema 2.1, segue que se v não é raiz, então existe apenas um vértice u do qual v é filho; u é então chamado *pai* de v. Se numa árvore T todos os nós têm no máximo dois filhos, diz-se que T é uma árvore *binária*. Na verdade, as relações de descendência entre os nós de uma árvore representam uma ordem, a *ordem em árvore*

em $V(T)$ associada a T e r: $u \leq v$, se $u \in rTv$. Pode-se ver que toda ordem em árvore é uma ordem parcial bem ordenada onde r é o elemento mínimo, toda folha da árvore é um elemento maximal, as extremidades de uma aresta são comparáveis e todo caminho de r a um nó v é uma cadeia, ou seja, um conjunto da forma $\{u : u \leq v\}$ onde todos os elementos são comparáveis entre si.

Propriedade Helly

Definição 2 (Propriedade Helly). Uma família $\{T_i\}_{i\in I}$ de subconjuntos de um conjunto T é dita como satisfazendo a *propriedade Helly* se $J \subseteq I$ e $T_i \cap T_j \neq \emptyset$ para todo $i, j \in J$ implica que $\bigcap_{j\in J} T_j \neq \emptyset$.

Lema 2.2. *Uma família $\{T_i\}_{i\in I}$ de sub-árvores de uma floresta T satisfaz a propriedade Helly.*

Prova. Suponha $T_i \cap T_j \neq \emptyset$ para todo $i, j \in J \subseteq I$. Considere três nós u, v e w em T. Seja S o conjunto de índices s tal que T_s contém pelo menos dois destes três nós, e sejam P_{uv}, P_{vw} e P_{uw} caminhos em T conectando u a v, v a w e u a w, respectivamente. Como T é uma árvore, segue que $P_{uv} \cap P_{vw} \cap P_{uw} \neq \emptyset$; mas cada T_s $(s \in S)$ contém um destes caminhos, e logo:

$$\bigcap_{s\in S} T_s \supseteq P_{uv} \cap P_{vw} \cap P_{uw} \neq \emptyset.$$

O lema segue por indução no tamanho de J. Suponha então que

$$[\forall i, j \in J : T_i \cap T_j \neq \emptyset] \implies \bigcap_{p\in J} T_p \neq \emptyset$$

vale para toda família J de sub-árvores de tamanho no máximo k. Considere agora uma nova família de sub-árvores $J' = \{T_1, \ldots, T_{k+1}\}$. Pela hipótese de indução, existem nós u, v e w tais que:

$$u \in \bigcap_{i=1}^{k} T_i, \quad v \in \bigcap_{i=2}^{k+1} T_i, \quad w \in T_1 \cap T_{k+1}.$$

Pode-se notar que cada sub-árvore T_i contém pelo menos dois nós entre u, v e w, e logo, pelo exposto acima, $\bigcap_{i=1}^{k+1} T_i \neq \emptyset$, o que conclui a prova. □

A relação entre árvores e a propriedade Helly permite inclusive uma caracterização daquelas em função desta:

Lema 2.3. *Um grafo G é uma árvore se e somente se toda família de caminhos em G satisfaz a propriedade Helly.*

Prova.
($\Rightarrow$) Suponha que G seja uma árvore, e seja então $\mathcal{P} = \{P_i\}_{i\in I}$ uma família de caminhos arbitrária em G. Sejam J_k os subconjuntos de índices de I com k elementos tais que

$P_i \cap P_j \neq \emptyset$, $i, j \in J_k$. Para $k = 2$, claramente $\bigcap_{q \in J_k} P_q \neq \emptyset$. Para $k = 3$, sejam P_1, P_2 e P_3 os elementos de J_3, e v_1, v_2 e v_3 nós pertencentes a $P_2 \cap P_3$, $P_1 \cap P_3$ e $P_1 \cap P_2$, respectivamente. Suponha que $P_1 \cap P_2 \cap P_3 = \emptyset$. Desta forma, pode-se ver que $\langle v_1 P_2 v_2 P_3 v_3 P_1 v_1 \rangle$ induz um ciclo em G, um absurdo, já que G é uma árvore; logo, $\bigcap_{q \in J_3} P_q \neq \emptyset$.

A prova da condição necessária segue por indução em k. Suponha então que $\bigcap_{q \in J_p} P_q \neq \emptyset$, para todo $J_p \subseteq I$, $p \leq k$, $k < |I|$, e seja P_{k+1} um caminho em $\mathcal{P}$ tal que $P_{k+1} \cap P_i \neq \emptyset$, para todo $P_i \in J_k$. Suponha agora por absurdo, que $\bigcap_{q \in J_k} P_q \cap P_{k+1} = \emptyset$. Sejam então $P_1, \ldots, P_k$ os caminhos representantes dos índices em J_k. Pela hipótese de indução, $\bigcap_{1 \leq q \leq k}^{q \neq i} P_q \cap P_{k+1} \neq \emptyset$ para $1 \leq i \leq k$, e sejam, portanto, v_i nós arbitrários em cada interseção $\bigcap_{1 \leq q \leq k}^{q \neq i} P_q \cap P_{k+1}$ e v_{k+1} um nó arbitrário em $\bigcap_{1 \leq q \leq k} P_q$. Se u e w são extremidades de P_{k+1}, suponha ainda, sem perda de generalidade, que os vértices u, v_1, $\ldots$, v_k, w aparecem nesta ordem em P_{k+1}. Basta ver agora que existem dois caminhos disjuntos unindo v_1 a v_k: um composto por $\langle v_1 P_{k+1} v_2 P_{k+1} \cdots P_{k+1} v_{k-1} P_{k+1} v_k \rangle$, e outro por $\langle v_1 P'_1 v_{k+1} P'_k v_k \rangle$, onde P'_1 e P'_k são quaisquer caminhos P_i, $1 \leq i \leq k$ com $i \neq 1$ e $i \neq k$, respectivamente. Logo, tem-se um absurdo pela existência de um ciclo unindo os $k + 1$ vértices v_i em G, o que leva a conclusão de que $\bigcap_{q \in J_k} P_q \cap P_{k+1} \neq \emptyset$. Como a propriedade Helly vale para qualquer família arbitrária de sub-árvores em G, equivale a dizer que esta vale para todas.

($\Leftarrow$) Considere $\mathcal{P}$ uma família qualquer de caminhos em um grafo $G = (V, E)$ que satisfaz a propriedade Helly, e seja $C = \langle u P_1 v P_2 w P_3 u \rangle$ um ciclo em G, com u, v, $w \in V$, e P_1, P_2, $P_3 \in \mathcal{P}$. Como $P_1 \cap P_2 = \{v\}$, $P_2 \cap P_3 = \{w\}$ e $P_1 \cap P_3 = \{u\}$, e pela hipótese de suficiência, deve-se ter $P_1 \cap P_2 \cap P_3 \neq \emptyset$, e logo C não pode existir. Como tal propriedade vale para todas as famílias de caminhos em G, conclui-se que G não contém ciclos, sendo uma árvore. □

2.4 Conectividade

A noção de conectividade em grafos já foi introduzida na Seção 2.2. No entanto, é necessário ainda definir um conceito para medição da conectividade, ou seja, de quão conexo um grafo pode ser.

Definição 3 (Grafo k-conexo e Conectividade). Um grafo $G = (V, E)$ é *k-conexo* (para $k \in \mathbb{N}$) se $|V| > k$ e $G[V \setminus X]$ é conexo para todo $X \subseteq V$, $|X| < k$. O maior inteiro k para o qual G é k-conexo é conhecido como a *conectividade* $\kappa(G)$ de G. Por convenção, $\kappa(K_n) = n - 1$, $n \geq 1$.

Em outras palavras, o conceito de conectividade implica que, em um grafo G, quaisquer dois vértices não são separados por menos de $\kappa(G)$ outros vértices.

Teorema 2.4 (Teorema de Menger). *Seja $G = (V, E)$ um grafo e A, $B \subseteq V$. O número mínimo de vértices separando A e B é igual ao número máximo de caminhos disjuntos entre A e B.*

Definição 4 (Grafo k-ligado). Um grafo G com pelo menos $2k$ vértices é dito k-ligado se, para quaisquer $2k$ vértices $s_1, \ldots, s_k$, $t_1, \ldots, t_k$ em G, este contém k caminhos disjuntos $P_i = \langle s_i \ldots t_i \rangle$, $1 \leq i \leq k$.

Vale observar que a definição de grafo k-ligado implica na satisfação do Teorema de Menger — e logo todo grafo k-ligado é também k-conexo — mas requer uma condição mais forte: não apenas dois conjuntos de vértices devem ser conectados por k caminhos disjuntos, mas devem existir caminhos distintos unindo cada par específico s_i-t_i. Desta forma, nem sempre a k-conectividade implica na k-ligação de um grafo. No entanto, uma relação pode ser estabelecida entre os dois conceitos:

Teorema 2.5 (Jung [1970]; Larman e Mani [1970]). *Existe uma função $f : \mathbb{N} \mapsto \mathbb{N}$ tal que todo grafo $f(k)$-conexo é k-ligado, para todo $k \in \mathbb{N}$.*

2.5 Emparelhamentos

Um *emparelhamento* em um grafo $G = (V, E)$ é um subconjunto $M \subseteq E$ de arestas tal que os elementos de M não são adjacentes entre si. Cada par de extremidades de cada aresta em M são ditas emparelhadas por M, e os vértices incidentes às arestas em M são ditos saturados, ou M-saturados. Um emparelhamento M é dito *maximal* em G se nenhuma outra aresta em G pode ser acrescentada a M de modo a aumentá-lo; um emparelhamento é dito *máximo* em G se não existe outro emparelhamento M' em G tal que $|M'| > |M|$.

2.6 Menores

Seja $e = (u, v)$ uma aresta de um grafo $G = (V, E)$. Denota-se por G/e o grafo obtido pela *contração* de e. A contração de e em G resulta em um novo vértice v_e em G/e, que é feito adjacente a todos os vizinhos de u e v em G. A definição formal segue.

Definição 5 (Contração). Uma *contração* (de arestas) é uma relação entre um grafo $G = (V, E)$, uma aresta $e = (u, v) \in E$ e um novo grafo $G' = (V', E')$, denotado por G/e, onde:

$$V' = V \setminus \{u, v\} \cup \{v_e\}$$

e

$$E' = \{(w, t) \in E : \{w, t\} \cap \{u, v\} = \emptyset\} \cup \{(v_e, w) : (u, w) \in E - e \text{ ou } (v, w) \in E - e\}$$

onde v_e é o novo vértice em G' correspondente a e em G.

O conceito de contração de arestas pode ser estendido para *contração de vértices*. Seja $U \subseteq V$ um subconjunto de vértices de G, definido acima. A contração (de vértices) de U em G é obtida pela contração de todas as arestas no conjunto $U_E = \{e = (u, v) : u, v \in U\}$. Claramente, como todos os vértices em U são contraídos através de arestas, U

induz um subgrafo conexo em G. O grafo obtido pela contração de U em G é denotado G/U. Vale notar que a contração de uma aresta $e = (u, v)$ pode ser vista como um caso especial de uma contração de vértices onde $U = \{u, v\}$. A contração de um par de vértices $\{u, v\}$, mesmo que não haja uma aresta entre eles, é também particularmente chamada de *identificação* dos vértices u e v em G; neste caso, o novo vértice obtido a partir da contração de $\{u, v\}$ é denotado por $u * v$.

Seja agora $\mathcal{U} = \{U_i\}_{1 \leq i \leq k}$, onde $U_i \subseteq V$ para todo $U_i \in \mathcal{U}$, uma família de subconjuntos de vértices de G tais que cada U_i induz um subgrafo conexo em G. Se o grafo X é obtido pela contração em G de todos os subconjuntos em $\mathcal{U}$ ($X = G/U_1/\cdots/U_i/\cdots/U_k$), então denota-se $G = MX$. Os elementos de $\mathcal{U}$ são chamados *ramos* de X.

Definição 6 (Menor de um grafo). Se um grafo $G = MX$ é um subgrafo de Y, então diz-se que X é um *menor* de Y, e denota-se por $X \preccurlyeq Y$.

Vale notar que, pela definição, todo subgrafo de um grafo é também seu menor (já que $G = MG$ para qualquer grafo G), assim como todo grafo é menor de si mesmo (já que todo grafo é subgrafo de si mesmo). Pode-se notar também que todos os menores de um grafo G podem ser obtidos a partir de uma série de operações de eliminação de vértices e arestas e contração de arestas, o que se pode concluir pelo fato de que a aplicação de qualquer uma destas operações a um grafo resulta em um menor deste, e pela transitividade da relação de menor. De fato, uma conclusão maior pode ser obtida:

Proposição 2.6 (Diestel [1999]). *A relação de menor $\preccurlyeq$ estabelece uma ordem parcial na classe de grafos finitos, ou seja, $\preccurlyeq$ é reflexiva, anti-simétrica e transitiva.*

Um conceito relacionado ao de menor é o de *menor topológico*. Seja X um grafo e G um grafo obtido de X pela substituição das arestas em X por caminhos independentes entre as extremidades de cada aresta. G é obtido por *subdivisão* (de arestas) de X, sendo denotado por $G = TX$. Analogamente, se $G = TX$ é um subgrafo de Y, X é dito um menor topológico de Y. A Proposição 2.6 também se aplica para a relação menor topológico, sendo estabelecida uma ordem parcial na classe de grafos finitos pela relação de menor topológico. Tal característica pode ser melhor aceita se a relação entre menor e menor topológico for melhor definida:

Proposição 2.7 (Diestel [1999]).

1. *Para um grafo X, todo TX é também um MX. Logo, qualquer menor topológico de um grafo é também seu menor (convencional).*
2. *Se $\Delta(X) \leq 3$, então qualquer MX contém um TX. Logo, qualquer menor com grau máximo no máximo 3 de um grafo é também um menor topológico deste.*

Classes de grafos

Uma *propriedade de grafos* P é uma função que mapeia cada grafo (no domínio de P) a um valor verdadeiro ou falso; assume-se também que P é invariante com relação a isomorfismo. Uma *classe de grafos* $\mathcal{C}$ é um conjunto de grafos tal que cada elemento de $\mathcal{C}$ satisfaz uma dada propriedade de grafos. Como exemplo, pode-se definir uma classe

$\mathcal{K}$ onde cada elemento K de $\mathcal{K}$ é tal que todos os vértices de K são adjacentes entre si; claramente, $\mathcal{K}$ é classe dos grafos completos.

Considere agora um conjunto ou classe de grafos $\mathcal{H}$. A classe

$$\text{Proib}_{\preccurlyeq}(\mathcal{H}) = \{G : H \not\preccurlyeq G \text{ para todo } H \in \mathcal{H}\}$$

dos grafos que não têm elementos de $\mathcal{H}$ como menores é uma classe fechada com relação a isomorfismo. A classe também pode ser caracterizada pelos grafos em $\mathcal{H}$, sendo estes conhecidos como *menores proibidos* (ou *excluídos*). Se $\mathcal{C} = \text{Proib}_{\preccurlyeq}(\mathcal{H})$ como acima, então diz-se também que $\mathcal{H}$ é o *conjunto de obstrução* da classe $\mathcal{C}$.

Pela Proposição 2.6, a classe $\text{Proib}_{\preccurlyeq}(\mathcal{H})$ é fechada também com relação a tomada de menores — se $G' \preccurlyeq G \in \text{Proib}_{\preccurlyeq}(\mathcal{H})$, então $G' \in \text{Proib}_{\preccurlyeq}(\mathcal{H})$. A proposição seguinte vincula as duas definições:

Proposição 2.8. *Uma classe de grafos $\mathcal{C}$ pode ser expressa por menores proibidos se e somente se $\mathcal{C}$ é fechada com relação a tomada de menores.*

Como Diestel (2000) define, uma classe fechada com relação a tomada de menores é também chamada *hereditária*.

2.7 Grafos Cordais

Um grafo não direcionado G é chamado *cordal* se cada ciclo de tamanho estritamente maior que 3 possui uma corda, ou seja, uma aresta unindo dois vértices não consecutivos do ciclo. Como G não possui subgrafos induzidos isomorfos a C_n, $n > 3$, o maior ciclo de G é um triângulo, e logo G é também chamado *triangulado*.

Uma forma simples de caracterizar grafos cordais é através de separadores minimais, como mostrado no lema a seguir.

Lema 2.9 (Dirac [1961]). *G é um grafo cordal se e somente se todo separador minimal induz um subgrafo completo em G.*

Prova.
($\Rightarrow$) Suponha que S seja um uv-separador minimal de $G = (V, E)$, sendo U e V as componentes de $G[V \setminus S]$ contendo u e v, respectivamente. Como S é minimal, todo $s \in S$ é adjacente pelo menos a algum vértice de U e a algum vértice de V. Logo, para qualquer par $x, y \in S$, existem caminhos $\langle x, u_1, \ldots, u_r, y\rangle$ e $\langle y, v_1, \ldots, v_t, x\rangle$, onde $u_i \in U, 1 \leq i \leq r$ e $v_i \in V, 1 \leq i \leq t$, que podem ser escolhidos de modo a ter o *menor* comprimento possível. Segue que $\langle x, u_1, \ldots, u_r, y, v_1, \ldots, v_t, x\rangle$ é um ciclo cujo comprimento é no mínimo 4, implicando que este deve ter uma corda. Mas $(u_i, v_j) \notin E$ pela definição de S, e $(u_i, u_j) \notin E$ e $(v_i, v_j) \notin E$, pela minimalidade de r e t. Logo a única corda possível é (x, y). Como isso vale para qualquer par $x, y \in S$, segue que S induz um subgrafo completo em G.

($\Leftarrow$) Seja $\langle u, x, v, y_1, y_2, \ldots, y_k, u\rangle$, $k \geq 1$, um ciclo de $G = (V, E)$. Como qualquer uv-separador minimal deve conter os vértices x e y_i, $1 \leq i \leq k$, segue que $(x, y_i) \in E$ e $(y_i, y_j) \in E$, e logo o ciclo sempre terá uma corda. □

Um outro resultado auxiliar envolvendo separadores minimais pode ser encontrado a partir da observação de que $r = t = 1$ na prova da condição necessária do lema acima, e logo, para qualquer par de vértices $x, y \in S$ existem vértices em U e V que são adjacentes tanto a x como a y. O resultado auxiliar segue de uma condição mais forte deste fato:

Lema 2.10. *Para qualquer separador minimal S de um grafo cordal $G = (V, E)$, existe um vértice v em cada componente conexa de $G[V \setminus S]$ tal que $S \subseteq N_G(v)$.*

Prova. Seja $G = (V, E)$ um grafo cordal e S um separador minimal de G. A prova é por indução no tamanho de S. Se $S = \{s\}$ então o lema é trivial. Seja então $S = \{s_1, \ldots, s_k\}$ e suponha que existe um vértice v tal que $S - s_k \subset N_G(v)$. Se $s_k \in N_G(v)$ a prova está completa. Suponha então o contrário, e logo deve existir um outro vértice u pertencente a mesma componente de v e adjacente a s_k (do contrário S não seria um separador minimal). Assuma ainda, sem perda de generalidade, que u é adjacente a v.

Pelo Lema 2.9, s_k é adjacente a todos os outros vértices de S. Para cada s_i, $1 \leq i \leq k-1$, $C_i = \langle s_i, v, u, s_k, s_i \rangle$ induz um ciclo em G, e logo, como G é cordal e $(v, s_k) \notin E$, deve existir uma corda unindo u e s_i. Portanto, $S \subseteq N_G(u)$ e a prova está concluída. □

Uma outra caracterização de grafos cordais, de aspecto mais algorítmico, envolve uma enumeração especial dos vértices do grafo. Para tanto, novos conceitos são necessários. Um vértice v de G é chamado *simplicial* se sua vizinhança $N_G(v)$ induz um subgrafo completo de G, ou seja, $N_G(v)$ é uma clique. Vértices simpliciais têm grande aplicação na elaboração de algoritmos para grafos cordais. Antes de apresentar uma caracterização algorítmica desta classe de grafos, um outro resultado auxiliar é necessário.

Lema 2.11 (Dirac [1961]). *Todo grafo cordal G tem um vértice simplicial. Além disso, se G não é completo, então ele tem dois vértices simpliciais não adjacentes.*

Prova. Se G é completo, o lema torna-se trivial. Suponha então que $G = (V, E)$ possui dois vértices u e v não adjacentes e que, por indução, o lema vale para todos os grafos com menor cardinalidade de vértices que G. Considere agora S um separador minimal para u e v, e as duas componentes (conexas) U e V de $G[V \setminus S]$ contendo u e v, respectivamente.

Pelo critério de indução, ou $G[U \cup S]$ tem dois vértices não adjacentes e simpliciais, dos quais um deve estar em U, já que S induz um subgrafo completo (Lema 2.9), ou $G[U \cup S]$ é completo e logo qualquer vértice deste é simplicial. Conseqüentemente, como $N_G(U) \subseteq U \cup S$, todo vértice simplicial em $G[U \cup S]$ o é também em G. Como o mesmo argumento aplica-se para V, G possui pelo dois vértices simpliciais, o que conclui a prova. □

Usando o resultado do Lema 2.11, Fulkerson e Gross (1965) *apud* (Golumbic 1980) sugeriram um procedimento iterativo para reconhecer grafos cordais: repetidamente localize um vértice simplicial e elimine-o do grafo até que ou nenhum vértice permaneça — e o grafo seja cordal — ou nenhum vértice seja simplicial, e logo o grafo não é cordal. Formalmente, a eliminação iterativa de vértices simpliciais pode ser representada por um esquema enumerativo.

Sejam então $G = (V, E)$ um grafo não direcionado e $\sigma = [v_1, v_2, \ldots, v_n]$ um ordenamento de vértices de V. Diz-se que σ é um *esquema perfeito de eliminação de vértices*,

ou simplesmente um *esquema perfeito*, se cada v_i é um vértice simplicial do subgrafo induzido $G[\{v_i, \ldots, v_n\}]$. O seguinte resultado formaliza o procedimento de Fulkerson e Gross:

Lema 2.12. *G é um grafo cordal se e somente se G admite um esquema de eliminação perfeito. Além disso, qualquer vértice simplicial pode iniciar um esquema perfeito.*

Prova.
($\Rightarrow$) Pelo Lema 2.12, se $G = (V, E)$ é cordal, então este possui um vértice simplicial x. Basta agora notar que, por indução, $G[V - x]$ é ainda cordal e deve admitir um esquema σ' de eliminação perfeito. Um esquema σ perfeito para G é simplesmente $\sigma = [x] \cup \sigma'$, ou seja, a junção de x a σ' como prefixo.

($\Leftarrow$) Sejam C um ciclo de G e x um vértice de C de menor índice em um esquema perfeito. Como $|N_G(x) \cap C| \geq 2$, a simplicialidade de x garante uma corda em C. □

Os grafos cordais possuem características especiais que serão melhor exploradas no próximo capítulo. E alguns casos, é interessante então obter um grafo cordal, supergrafo de um grafo original, acrescentando arestas a este. Um grafo $H = (V, E')$ é dito então uma *cordalização* de um grafo $G = (V, E)$ se G é um subgrafo gerador de H e H é cordal; neste caso, $E' \setminus E$ é uma *triangulação* de G resultando em H. H é dito uma k-cordalização de G se é uma cordalização cujo tamanho da maior clique é no máximo k.

k-árvores

Definição 7 (k-árvore). Uma *k-árvore* é definida recursivamente como se segue: uma *k-árvore* com $k + 1$ vértices consiste de uma clique com $k + 1$ vértices ($k + 1$-clique); dada uma *k-árvore* T_n com n vértices, $n > k + 1$, constrói-se uma *k-árvore* com $n + 1$ vértices acrescentando um novo vértice v a T_n e fazendo-o adjacente a cada vértice de uma k-clique de T_n e não adjacente aos $n - k$ vértices restantes. Uma *k-árvore parcial* é um subgrafo de uma k-árvore.

Lema 2.13. *Uma k-árvore é um grafo cordal.*

Prova. Seja T_n uma k-árvore com n vértices. Pela definição de k-árvore, sempre existe um vértice v_1 adjacente a uma k-clique, e logo, simplicial. Com a retirada de v_1 de T_n, tem-se ainda uma k-árvore T_{n-1} com $n - 1$ vértices, de onde se pode retirar outro vértice v_2 simplicial (e adjacente a outra k-clique). Pode-se perceber então que $n - k$ vértices simpliciais podem ser retirados de T_n de forma iterada. Como os outros k vértices restantes são simpliciais, tem-se que $\sigma = [v_1, v_2, \ldots, v_n]$ constitui um esquema perfeito de eliminação, e logo T_n é cordal. □

2.8 Lógica Monádica de Segunda Ordem

Definição 8 (Grafo terminal). Um *grafo terminal* é uma tripla (V, E, X), onde (V, E) é um grafo simples e $X \subseteq V$ é um conjunto ordenado de l vértices, $0 \leq l \leq |V|$, denotado

por $X = [x_1, \ldots, x_l]$. Os vértices em X são denominados *terminais*, e os vértices em $V \setminus X$, *internos*.

Um grafo terminal $G = (V, E, X)$ com l terminais é também chamado l-*terminal*, e $x_i \in X$, $1 \leq i \leq l$, é denominado o i-ésimo terminal de G. O conjunto de vértices terminais de G é denotado por $\text{Term}(G) = X$ — e logo, se G é l-terminal, $|\text{Term}(G)| = l$ — e o conjunto de vértices internos de G é denotado por $\text{Int}(G) = V \setminus X$. Um grafo terminal G é dito *aberto* se não existem arestas entre os terminais de G. De modo semelhante a grafos convencionais, diz-se que dois grafos terminais $G_1 = (V_1, E_1, [x_1, \ldots, x_l])$ e $G_2 = (V_2, E_2, [y_1, \ldots, y_p])$ são *isomorfos* se $l = p$ e se existe um isomorfismo entre $G'_1 = (V_1, E_1)$ e $G'_2 = (V_2, E_2)$ mapeando cada x_i a y_i, $1 \leq i \leq l$. O isomorfismo entre grafos terminais é denotado da mesma forma que em grafos convencionais, ou seja, se G_1 e G_2 são isomorfos, escreve-se $G_1 \approx G_2$.

Definição 9 (Operação de junção terminal). A *operação de junção terminal* $\oplus$ mapeia dois grafos terminais G e H com o mesmo número de terminais em um grafo simples $G \oplus H$ tomando a união de G e H e identificando os terminais correspondentes, ou seja, identificando o i-ésimo terminal de G com o i-ésimo terminal de H e removendo arestas múltiplas.

Definição 10 (Propriedade de índice finito). Seja P uma propriedade de grafos, e l um número natural. Para grafos l-terminais G_1 e G_2, define-se uma relação de equivalência $\sim_{P,l}$ da seguinte forma:

$$G_1 \sim_{P,l} G_2 \quad \Leftrightarrow \quad \text{para todo grafo } l\text{-terminal } H \ : \ P(G_1 \oplus H) \Leftrightarrow P(G_2 \oplus H).$$

A propriedade P é dita *de índice finito* se, para todo l, $\sim_{P,l}$ tem finitas classes de equivalência.

De acordo com Courcelle (1990), uma classe de grafos cuja propriedade é de índice finito é chamada de *estado finito*. Ele definiu uma ampla classe de grafos de estado finito. Nos próximos capítulos a aplicação desse tipo de classe ficará mais clara, assim como a sua grande vantagem computacional. Courcelle (1990) definiu uma linguagem para caracterização de tais classes:

Definição 11 (Lógica Monádica de Segunda Ordem para grafos). A *Lógica Monádica de Segunda Ordem* (LMSO) para grafos $G = (V, E)$ consiste de uma linguagem na qual os predicados podem ser construídos usando:

1. Os conectivos lógicos $\wedge$ (e), $\vee$ (ou), $\neg$ (negação), $\implies$ (implica) e $\Leftrightarrow$ (se e somente se).
2. Variáveis que podem ser variáveis de vértices (com domínio V), de arestas (com domínio E), de conjuntos de vértices (com domínio $\mathcal{P}(V)$, a família de todos os subconjuntos de V), e de conjuntos de arestas (com domínio $\mathcal{P}(E)$, a família de todos os subconjuntos de E).
3. Os quantificadores existencial ($\exists$) e universal ($\forall$).
4. As seguintes relações binárias:

- $v \in W$, onde v e W são variáveis de vértice e conjunto de vértices, respectivamente;
- $e \in F$, onde e e F são variáveis de aresta e conjunto de arestas, respectivamente;
- "v e w são adjacentes em G", onde v e w são variáveis de vértices;
- "v é incidente a e em G", onde v e e são variáveis de vértice e aresta, respectivamente;
- igualdade entre variáveis do mesmo tipo.

Seja R um predicado LMSO tal que R não contém variáveis livres. Então diz-se que um grafo G *satisfaz* R se R resulta em verdadeiro para G. Um propriedade de grafos P é *MS-definível* se existe um predicado R definido em LMSO para grafos sem variáveis livres tal que, para todo grafo G, $P(G)$ vale se e somente se G satisfaz R. Uma classe de grafos é MS-definível se sua propriedade é MS-definível.

Como exemplo, a fim de esclarecer melhor o conceito de MS-definibilidade, pode-se mostrar que a classe de grafos bipartites é MS-definível. Para tanto, considere o seguinte predicado LMSO R:

$$\begin{aligned} R \;=\; & \exists U \exists W \, (U \subseteq V) \wedge (W \subseteq V) \wedge (U \cap W = \emptyset) \wedge (U \cup W = V) \\ & \wedge \Big(\forall u \forall v \, \Big((u \in V) \wedge (v \in V) \wedge (u \text{ e } v \text{ são adjacentes}) \Big) \implies \\ & \Big((u \in U) \Leftrightarrow (v \in W) \Big) \Big) . \end{aligned}$$

Vale notar que as operações de continência, interseção e união entre conjuntos não são definidas na linguagem, e constam na expressão de R acima apenas por simplicidade. No entanto, pode-se definir $U \subseteq W$ como $\forall u (u \in U) \implies (u \in W)$, e expressar:

$$U \cap W = \emptyset \quad \text{por} \quad \forall u \Big((u \in U) \vee (u \in W) \Big) \implies \neg \Big((u \in U) \wedge (u \in W) \Big)$$

e

$$U \cup W = V \quad \text{por} \quad \forall u \Big((u \in U) \vee (u \in W) \Big) \implies (u \in V).$$

Na expressão de R, pode-se ver que U e W representam classes de uma partição de V, e logo, G é bipartite se e somente se G satisfaz R. Como conseqüência direta, a propriedade P, "ser bipartite", é MS-definível. Courcelle (1990) mostrou que toda propriedade MS-definível caracteriza uma classe de grafos de estado finito.

A definição de propriedade de grafos pode ainda ser estendida de modo a envolver outras variáveis. Seja então P uma propriedade de grafos *estendida* com variáveis G, $X_1, \ldots, X_r$, onde G é um grafo e, para cada i, $1 \leq i \leq r$, $X_i \in D_i$ para algum domínio D_i. Então, analogamente, diz-se que $P(G, X_1, \ldots, X_r)$ é MS-definível se existe um predicado LMSO para grafos $R(Y_1, \ldots, Y_r)$, com variáveis livres $Y_1, \ldots, Y_r$, tal que, para todo grafo G e variáveis $X_1, \ldots, X_r$, P vale se e somente se G satisfaz $R(X_1, \ldots, X_r)$.

Como exemplo, considere agora X_1 e X_2 subconjuntos de vértices de um grafo G. A propriedade "ser bipartite com classes X_1 e X_2" para G, $P(G, X_1, X_2)$, é então MS-definível se $P(G, X_1, X_2)$ vale se e somente se G satisfaz um predicado LMSO com parâmetros Y_1 e Y_2. Seja $R(Y_1, Y_2)$ o predicado LMSO definido por:

$$\begin{aligned} R(Y_1, Y_2) \quad = \quad & (Y_1 \cap Y_2 = \emptyset) \wedge (Y_1 \cup Y_2 = V(G)) \wedge \\ & \Big(\forall u \forall v \ \Big((u \in V(G)) \wedge (v \in V(G)) \wedge (u \text{ e } v \text{ são adjacentes})\Big) \implies \\ & \Big((u \in Y_1) \Leftrightarrow (v \in Y_2)\Big)\Big) . \end{aligned}$$

Pode-se ver que, para quaisquer G, X_1 e X_2 tais que G é bipartite com classes X_1 e X_2, G satisfaz $R(X_1, X_2)$. Logo, $P(G, X_1, X_2)$ vale, e é MS-definível.

Capítulo 3

Decomposição em Árvore

3.1 Introdução

Muitos problemas de otimização combinatória podem ser modelados e, conseqüentemente, tratados por grafos. Dentre as diversas vantagens desta abordagem, uma das principais reside no fato de que, para determinadas classes de grafos, algoritmos específicos podem ser projetados de modo a resolver o problema com complexidade computacional bem menor, polinomial. Ou seja, como a complexidade destes problemas é exponencial, a restrição do problema a certos tipos de grafos o torna tratável praticamente.

Dentre as classes de grafos empregadas para tal estratégia, destacam-se as *árvores*. Uma árvore não possui ciclos, e logo tal característica mostra-se extremamente favorável para aplicação de algoritmos recursivos, por exemplo. Basta notar, por exemplo, que diversas estratégias gerais de resolução buscam o espaço viável de soluções através de uma árvore, como no caso da programação dinâmica e da divisão e conquista; além disso, algoritmos comuns de busca em grafos, como busca em largura e busca em profundidade, percorrem o grafo formando uma árvore.

Infelizmente, a maior parte dos problemas não são modelados em árvores, o que inviabiliza tal abordagem. Uma estratégia interessante seria então estender o tratamento natural dado às árvores para outras classes de grafos, e, finalmente, para todos os grafos. Claramente, é esperado que, na medida em que o grafo seja menos semelhante a uma árvore, a complexidade de um algoritmo para este grafo seja maior, mais próxima da exponencial; por outro lado, a complexidade poderá ser polinomial, ou mesmo linear, quão mais similar a uma árvore o grafo seja.

A *Decomposição em Árvore* é uma abordagem que tem essa finalidade: estabelecer relações estruturais num grafo de forma que este assemelhe-se a uma árvore. Constitui-se da relação entre subgrafos de um grafo G e nós de uma árvore de modo que o conjunto de arestas de G seja particionado numa estrutura em árvore. O conceito formal foi apresentado por (Robertson e Seymour 1986):

Definição 12 (Decomposição em árvore). Uma *decomposição em árvore (DEA)* D de um grafo $G = (V, E)$ é um par $(\mathcal{X} = \{X_i : i \in I\}, T = (I, F))$, sendo $\{X_i : i \in I\}$ uma família de subconjuntos de V, um para cada nó de T, e T uma árvore, tais que:

[P1] $\bigcup_{i \in I} X_i = V$;

[P2] para toda aresta $(v, w) \in E$, existe um $i \in I$ tal que $v \in X_i$ e $w \in X_i$;

[P3] para todo $i, j, k \in I$, se j está no caminho entre i e k em T, então $X_i \cap X_k \subseteq X_j$.

A família $\mathcal{X}$ compreende as *partes* da decomposição em árvore, ou seja, cada X_i, $i \in I$. As duas primeiras restrições implicam que $\mathcal{X}$ compreende uma família de subgrafos de G — sendo cada subgrafo obtido pela indução de cada parte em G — enquanto que a última restrição garante a estrutura em árvore da decomposição. Isto pode não ser imediato, então, por enquanto, pode-se substituir [P3] por uma restrição equivalente:

[P3'] para todo $v \in V$, $T_v = \{i \in I : v \in X_i\}$ induz uma árvore.

Como cada vértice de um grafo é relacionado a uma árvore, claramente a união de todos os vértices resulta numa árvore. Embora esta propriedade de estrutura em árvore seja mais clara quando [P3'] é considerada, [P3] é bem mais útil e direta quando se deseja construir decomposições em árvore. A equivalência entre as duas restrições é mostrada a seguir.

Lema 3.1. *As restrições* [P3] *e* [P3'] *da definição de decomposição em árvore são equivalentes.*

Prova.
([P3'] ⇒ [P3]). Sejam $i, j \in I$ nós de T. Se $X_i \cap X_j = \emptyset$, então [P3] segue, já que qualquer X_k, $k \in iTj$, contém a interseção entre X_i e X_j. Suponha então que $X_i \cap X_j \neq \emptyset$, e seja $v \in X_i \cap X_j$. Claramente, por [P3'], a árvore T_v contém o caminho iTj. Logo, para todo $k \in iTj$, segue que $v \in X_k$. Pelo mesmo motivo, todo vértice pertencente a interseção entre X_i e X_j pertence a todo X_k, $k \in iTj$, e logo $X_k \supseteq X_i \cap X_j$.

([P3] ⇒ [P3']). Seja $v \in V$ e $T_v = \{i \in I : v \in X_i\}$. Se $|T_v| = 1$, então T_v é um grafo trivial, e logo uma árvore; se $T_v = \{i, j\}$, então, por [P3], e como $X_i \cap X_j \neq \emptyset$, existe uma aresta entre i e j, e logo T_v é uma árvore. Suponha então que $|T_v| \geq 3$, e sejam i, j e k vértices em T_v. Pela definição de T_v, $v \in X_i \cap X_j \cap X_k$, e seja X_p, $p \in T_v$, tal que $X_p \supseteq X_i \cap X_j \cap X_k$. Segue então por [P3] que quaisquer caminhos unindo nós diferentes de $\{i, j, k\}$ se interceptam em p. Logo, qualquer família de caminhos em T_v satisfaz a propriedade Helly, e, pelo Lema 2.3, T_v é uma árvore. □

Na verdade, para cada $v \in V$, T_v induz uma *sub-árvore* de T. Formalmente, denota-se então por T_v a sub-árvore de T induzida por $v \in V$ em D, $T_v = \{i \in I : v \in X_i\}$.

Pode-se perceber também que, a partir da definição, várias decomposições podem ser obtidas para o mesmo grafo. A Figura 3.1 ilustra algumas decomposições em árvore. Pode-se ver que estas variam de acordo com o tamanho da árvore T e dos subconjuntos X_i. O tamanho da árvore pode não ser tão decisivo para julgar a estrutura em árvore do

grafo, já que a Definição 12 permite a inclusão de subconjuntos vazios, e logo permite também a inclusão de nós redundantes na árvore.

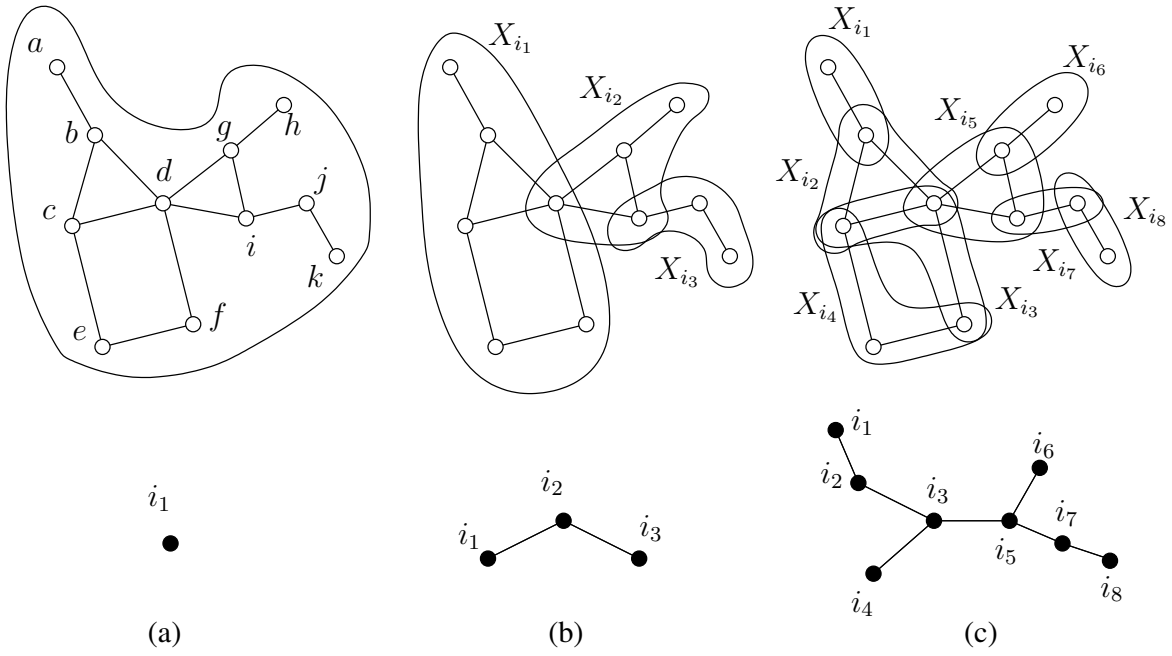


Figura 3.1. Exemplos de decomposição em árvore.

Claramente todo grafo G admite uma decomposição *trivial* em árvore, $(\mathcal{X} = \{V\}, T = (\{i\}, \emptyset))$, como a decomposição ilustrada na Figura 3.1(a). Além disso, é possível a existência de dois nós adjacentes i e j em uma decomposição em árvore D para G tais que $X_i \subseteq X_j$. Pode-se ver que X_i é redundante, pois pode-se obter uma nova decomposição em árvore D' a partir da identificação de i e j em T. Por outro lado, se uma decomposição em árvore D é tal que quaisquer nós adjacentes i e j satisfazem $X_i \not\subseteq X_j$ e $X_j \not\subseteq X_i$, então D é dita uma *boa* decomposição em árvore. Decomposições triviais e não boas são muito pouco usadas na prática, tendo mais aplicação como base de indução ou construções usadas nas provas de alguns resultados. Suas presenças neste texto estão limitadas a tais situações, sendo as decomposições boas bem mais úteis.

Mesmo considerando decomposições boas, o tamanho das partes pode, no entanto, ser ainda significativo ao se avaliar a similaridade do grafo com uma árvore: quão menores os tamanhos dos subconjuntos X_i mais semelhante é o grafo a uma árvore. O efeito dos tamanhos das partes fica mais evidente ao se comparar as Figuras 3.1(b) e 3.1(c). Deste modo, uma medida quantitativa de similaridade se faz desejável, e pode agora ser definida.

Definição 13 (Largura em árvore). A *largura em árvore de uma decomposição* $D = (\mathcal{X} = \{X_i : i \in I\}, T = (I, F))$ é definida como $LA_G(D) = \max_{i \in I}\{|X_i| - 1\}$. A *largura em árvore de um grafo* G é a largura *mínima* dentre todas as decomposições em árvore possíveis de G, ou seja, $LA(G) = \min\{LA_G(D) : D$ é uma decomposição em árvore de $G\}$.

A menos que haja confusão na notação, pode-se também referir à largura de uma decomposição em árvore D de um grafo G simplesmente como $LA(D)$. Como exemplo, as larguras em árvore das decomposições das Figuras 3.1(a), 3.1(b) e 3.1(c) são, respectivamente, 10, 5 e 2. A largura em árvore de um grafo procura estabelecer uma medida de similaridade a uma árvore para o grafo. Se a largura em árvore de um grafo G é pequena, então pode-se construir uma decomposição em árvore para G com partes pequenas, e logo G se assemelhará grosseiramente a uma árvore. Na verdade, se $LA(G)$ for pequeno o suficiente, G pode inclusive ser uma árvore. Uma decomposição em árvore D de um grafo G que realiza a largura de G, ou seja, tal que $LA_G(D) = LA(G)$, é dita *mínima*, ou *ótima*.

De modo a simplificar a notação ao longo do texto, $(\mathcal{X}, T)$ será usado como simplificação para $(\mathcal{X} = \{X_i : i \in I\}, T = (I, F))$ sempre que não ocorrer risco de ambigüidade. Nas próximas seções deste capítulo serão abordadas algumas características de grafos com largura em árvore pequena, e outras que impedem alguns grafos de terem largura limitada. Antes, no entanto, algumas propriedades gerais de decomposições em árvore são úteis, e algumas vezes necessárias, para melhor compreensão do que virá.

3.2 Algumas propriedades de DEA

A primeira propriedade de uma decomposição em árvore $D = (\mathcal{X}, T)$ de um grafo G faz com que G herde a conectividade minimal de T pelas partes de D: assim como cada aresta (i, j) é uma ponte em T, $X_i \cap X_j$ é um corte em G.

Propriedade 1. Sejam $G = (V, E)$ um grafo e $D = (\mathcal{X} = \{X_i : i \in I\}, T = (I, F))$ uma decomposição em árvore de G. Seja $e = (i, j)$ uma aresta qualquer de T, e sejam T_i e T_j as componentes de $T - e$ contendo i e j, respectivamente. Então $X_i \cap X_j$ separa $U_i = \bigcup_{t \in T_i} X_t$ de $U_j = \bigcup_{t' \in T_j} X_{t'}$ em G.

Prova. Como (i, j) é uma ponte em T, então qualquer caminho unindo $t \in T_i$ e $t' \in T_j$ passa por (i, j). Portanto, pela definição de decomposição em árvore, $U_i \cap U_j \subseteq X_i \cap X_j$. Basta mostrar agora que não existem arestas (u_i, u_j) com $u_i \in U_i$ e $u_j \in U_j$ para que $X_i \cap X_j$ seja de fato um U_iU_j-separador. Suponha então que exista uma aresta (u_i, u_j). Pela definição de DEA, deve existir um $t \in T$ tal que $u_i, u_j \in X_t$. Mas, pela escolha de u_i e u_j, t não pode pertencer nem a T_i nem a T_j, caso contrário existiria um ciclo em T, um absurdo. A Figura 3.2 ilustra a propriedade. □

Ao se considerar a largura em árvore de um grafo G, é natural questionar qual a sua influência nas larguras em árvore de grafos derivados de G e com menos vértices que G. As propriedades seguintes abordam tais relações entre G e seus subgrafos, componentes e menores.

Propriedade 2. Seja G um grafo e H um subgrafo de G. Então $LA(H) \leq LA(G)$.

Prova. Seja $D = (\mathcal{X}, T = (I, F))$ uma decomposição em árvore que realiza a largura em árvore de G — ou seja, D tem largura mínima. Considere agora a família $\mathcal{W} = \{W = X \setminus (V(G) \setminus V(H)) : X \in \mathcal{X}\}$. Claramente, como $\mathcal{X}$ cobre $V(G)$, assim também $\mathcal{W}$ cobre $V(H)$, e logo, como D é uma decomposição em árvore de G, $D' = (\mathcal{W}, T)$ é

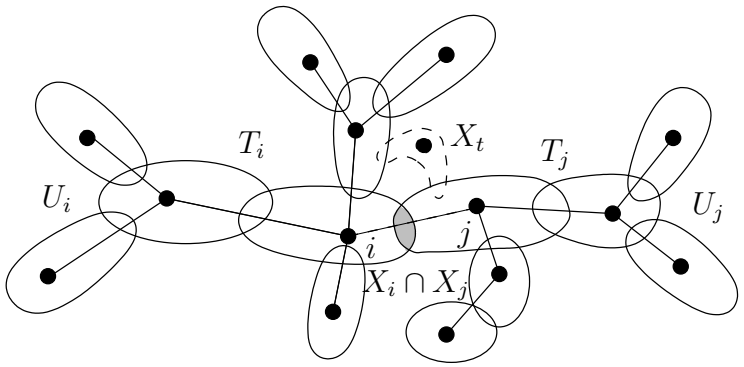


Figura 3.2. Uma aresta (i, j) de uma decomposição em árvore de um grafo G corresponde a um separador em G.

uma decomposição em árvore de H (D' é também chamada uma *subdecomposição* em árvore de G limitada a H). Como para todo $i \in I$, $W_i \subseteq X_i$, $LA(D') \leq LA(D)$, e conseqüentemente, pela minimalidade de D, $LA(H) \leq LA(G)$. A nova decomposição em árvore pode ser vista na Figura 3.3. □

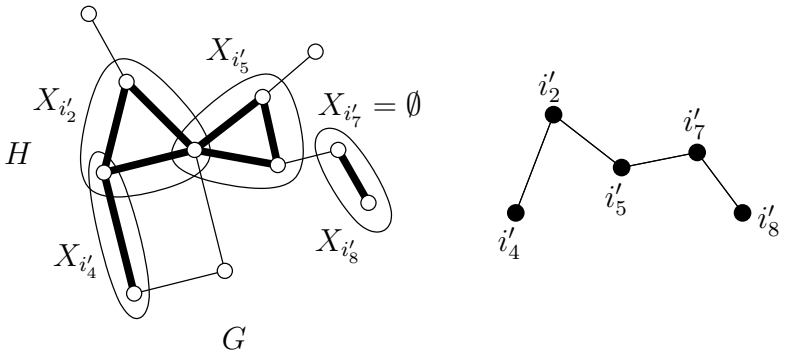


Figura 3.3. Decomposição em árvore de um subgrafo H de G, destacado por arestas em negrito.

Propriedade 3. A largura em árvore de um grafo G é a máxima largura em árvore das componentes de G.

Prova. Sejam $C_1, \ldots, C_r$ as r componentes de $G = (V, E)$, e $D_1, \ldots, D_r$ decomposições em árvore de largura mínima de cada componente, respectivamente. Para cada decomposição $D_j = \{\mathcal{X}_j, T_j = (I_j, F_j)\}$, selecione aleatoriamente um vértice $i_j \in I_j$. Considere agora a decomposição em árvore obtida pela união das partes em cada $\mathcal{X}_j$ e pela adição de uma parte $X^* = \emptyset$ com um nó correspondente i^* adjacente a cada i_j, ou seja, $D^* = (\mathcal{X}^*, T^* = (I^*, F^*))$, onde:

$$\mathcal{X}^* = \left(\bigcup_{j=1}^{r} \mathcal{X}_j \right) \cup \{X^*\},$$

$$I^* = \left(\bigcup_{j=1}^{r} I_j \right) \cup \{i^*\},$$

$$F^* = \left(\bigcup_{j=1}^{r} F_j \right) \cup \{(i^*, i_j) : i_j \text{ algum vértice em } I_j, 1 \leq j \leq r\}).$$

A Figura 3.4 ilustra tal decomposição. A largura de D^* pode ser então facilmente encontrada:

$$LA(D^*) = \max_{i \in I^*}\{|X_i| - 1\} = \max_{1 \leq j \leq r}\{\max_{i \in I_j}\{|X_i| - 1\}\} = \max_{1 \leq j \leq r}\{LA(D_j)\}$$

e logo, pela minimalidade de cada D_j, $LA(G) = \max_{1 \leq j \leq r}\{LA(C_j)\}$, e o lema segue. □

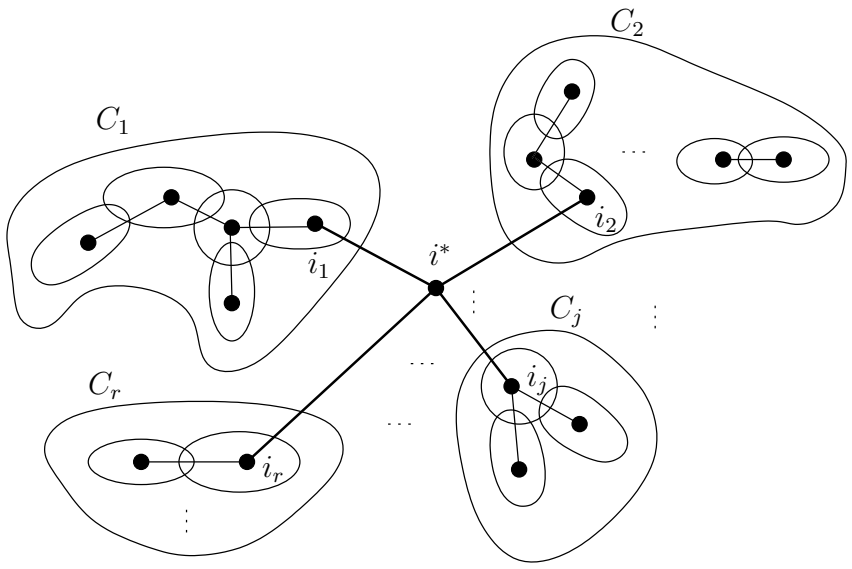


Figura 3.4. Decomposição em árvore de um grafo usando suas componentes.

Propriedade 4. Seja G um grafo e H um menor de G. Então $LA(H) \leq LA(G)$.

Prova. Por definição, H pode ser obtido de G por uma série de contrações ou retiradas de arestas, ou eliminações de vértices. Se apenas as duas últimas operações foram aplicadas, H é um subgrafo de G, e, pela Propriedade 2, o resultado segue. Basta então mostrar que a largura em árvore de um grafo não aumenta com a contração de arestas deste.

Seja então $D = (\mathcal{X}, T)$ uma decomposição em árvore de G. Considere então $e = (u, v)$ uma aresta de G a ser contraída como operação para obtenção de H, e seja w o novo vértice adicionado a G', o grafo obtido de G pela contração de e. Uma nova decomposição em árvore $D' = (\mathcal{X}', T)$ pode então ser definida, onde:

$$\mathcal{X}' = \{X \setminus \{u, v\} \cup \{w\} : X \in \mathcal{X}, u \in X \text{ ou } v \in X\} \cup \{X : X \in \mathcal{X}, u \notin X \text{ e } v \notin X\}.$$

Claramente, como H tem menos vértices que G, $LA_H(D') \leq LA_G(D)$. Como a desigualdade aplica-se a qualquer decomposição em árvore e qualquer menor de G, em particular deve valer também para a decomposição em árvore mínima de G, e logo segue que $LA(H) \leq LA(G)$, H um menor de G. Na Figura 3.5 pode-se ver uma decomposição em árvore de um menor de um grafo G, obtido pela contração das arestas em negrito. □

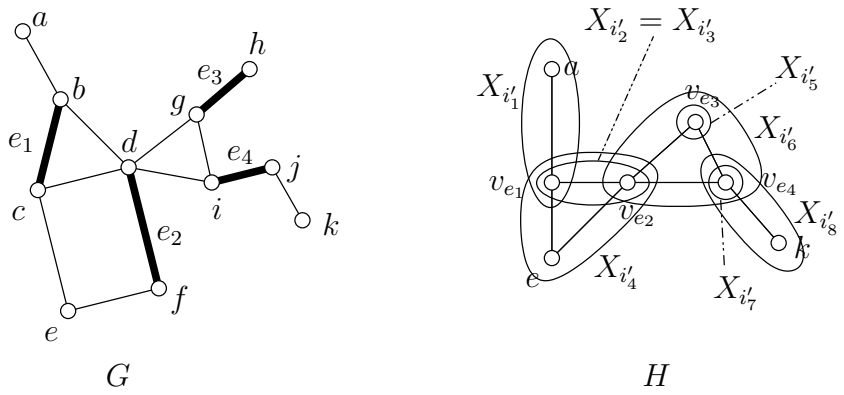


Figura 3.5. Decomposição em árvore de um menor H de um grafo G, obtido pela contração das arestas em negrito.

A vantagem no uso das três propriedades acima é que, além da determinação de um limitante superior para a largura em árvore de um grafo obtido a partir de outro — por operações de subgrafo, menor e componentes — as decomposições em árvore dos grafos resultantes podem também ser obtidas, como mostram as provas das propriedades. Além disso, pode-se de fato obter uma decomposição em árvore para qualquer grafo a partir de um menor deste por contração, como mostra o lema seguinte.

Lema 3.2. *Sejam $G = (V, E)$ um grafo, H um menor de G pela contração dos subconjuntos disjuntos de vértices $U_1, \ldots, U_r$ em G e $D = (\mathcal{X}, T)$ uma decomposição em árvore de H. Seja $f : V(H) \mapsto \mathcal{P}(V(G))$ uma função tal que $f(v) = \{v\}$ se $v \notin \bigcup_{i=1}^{r} U_i$ e $f(v) = U_i$ se $v \in U_i$ para algum i, $1 \leq i \leq r$. Então*

$$D' = (\mathcal{W} = \{W_i : W_i = \bigcup_{v \in X_i} f(v),\, X_i \in \mathcal{X}\}, T)$$

é uma decomposição em árvore para G.

Prova. Sejam G, H, $U_1, \ldots, U_r$, D e D' como no enunciado. Pode-se ver que D' satisfaz [P1] da definição de decomposição em árvore, já que

$$\bigcup_{i \in I} W_i = \bigcup_{i \in I} \bigcup_{v \in X_i} f(v) = V$$

ou seja, a união das partes corresponde à união dos vértices em cada U_j, $1 \leq j \leq r$, e dos vértices em G não inclusos em alguma parte, e que correspondem, portanto, ao conjunto de vértices de G.

De modo semelhante, pode-se dividir as arestas de G em quatro classes, de acordo com suas extremidades: uma aresta $e = (u, v)$ pode ser tal que (1) u e v não estejam em nenhum U_j, $u, v \notin \bigcup_{1 \leq j \leq r} U_j$; (2) exatamente uma das extremidades esteja em algum U_j; (3) as extremidades pertencem a conjuntos diferentes, $u \in U_i$, $v \in U_j$, onde $1 \leq i \neq j \leq r$; ou (4) ambas as extremidades pertencem a um mesmo U_j, $u, v \in \bigcup_{1 \leq j \leq r} U_j$. Sejam agora v_{U_i} os vértices em H correspondentes a contração de U_i em G. Basta agora ver que as arestas $e = (u, v)$ em G dos tipos (1), (2) e (3) correspondem a arestas em H dos tipos (1) $e' = (u, v)$; (2) $e' = (u, v_{U_i})$, onde somente v pertence a algum U_i; e (3) $e' = (v_{U_i}, v_{U_j})$, onde $u \in U_i$ e $v \in U_j$, $U_i \neq U_j$. As arestas em G do último tipo são contraídas em H. Como as arestas dos três primeiros tipos estão cobertas por partes em D, estas estão cobertas também por partes em D', já que para todo $X \in \mathcal{X}$ existe um $W \in \mathcal{W}$ tal que $X \subseteq W$. As arestas em G do terceiro tipo, com ambas as extremidades pertencentes a algum U_i, também estão cobertas em D', já que v_{U_i} pertence a algum $X \in \mathcal{X}$. Logo, a propriedade [P2] é também satisfeita. A Figura 3.6 permite uma melhor compreensão da satisfação das propriedades.

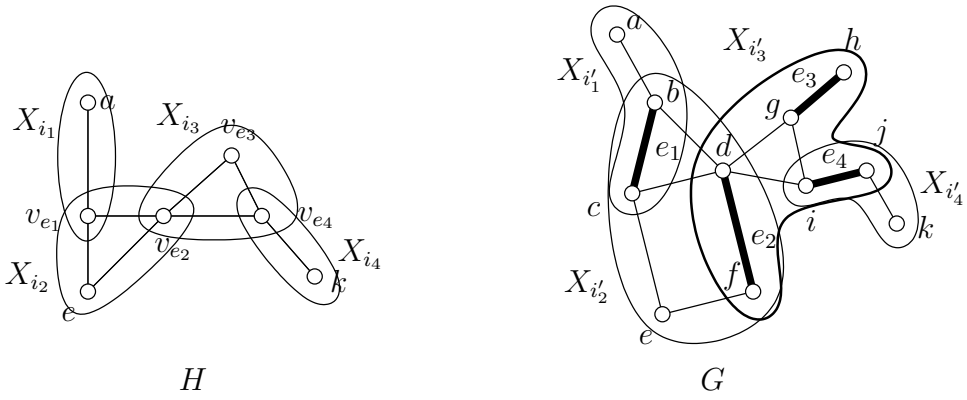


Figura 3.6. Decomposição em árvore de um grafo G a partir de um menor H por contração. Os conjuntos de contração correspondem às extremidades das arestas em negrito.

Para concluir a prova, resta somente satisfazer [P3]. Sejam então i, j e k vértices em T tais que $k \in iTj$, e logo, por [P3], $X_i \cap X_j \subseteq X_k$. De modo a facilitar a notação, considere $f(S)$, S um conjunto, como sendo $f(S) = \bigcup_{s \in S} f(s)$. Claramente, $f(X) = W$, $X \in \mathcal{X}$, $W \in \mathcal{W}$, por esta definição. Pode-se perceber que f preserva interseção entre conjuntos, ou seja:

$$f(A \cap B) = \bigcup_{s \in A \cap B} f(s) = \left(\bigcup_{s \in A} f(s) \right) \bigcap \left(\bigcup_{s \in B} f(s) \right) = f(A) \cap f(B)$$

e logo, para conjuntos A, B e C, $A \cap B = C$ se e somente se $f(A) \cap f(B) = f(A \cap B) = f(C)$. Mas, como para dois conjuntos A e B, $A \subseteq B$ se e somente se $A \cap B = A$, tem-se também que $A \subseteq B$ se e somente se $f(A) \subseteq f(B)$, já que $A \cap B = A$ equivale a $f(A \cap B) = f(A)$.

Portanto, $X_i \cap X_j \subseteq X_k$ em D equivale a

$$f(X_i \cap X_j) \subseteq f(X_k) \Leftrightarrow f(X_i) \cap f(X_j) \subseteq f(X_k) \Leftrightarrow W_i \cap W_j \subseteq W_k$$

o que equivale a satisfação de [P3] em D', e logo D' é uma decomposição em árvore de G. □

Com a ajuda das propriedades aqui demonstradas, pode-se agora dispor de mais ferramentas para caracterização de grafos com largura em árvore limitada. Inicialmente, é interessante estudar grafos com largura em árvore pequena. Se G é um grafo com pelo menos uma aresta, então claramente $LA(G) \geq 0$. Com exceção de tais casos extremamente raros, a menor largura em árvore de um grafo é 1; tais grafos são analisados na próxima seção.

3.3 DEA para florestas

Se um grafo G é uma floresta, é natural intuir que a largura em árvore de G é muito pequena, já que cada componente de G é, por si, uma árvore:

Teorema 3.3. *Um grafo G é uma floresta se e somente se G tem largura em árvore 1.*

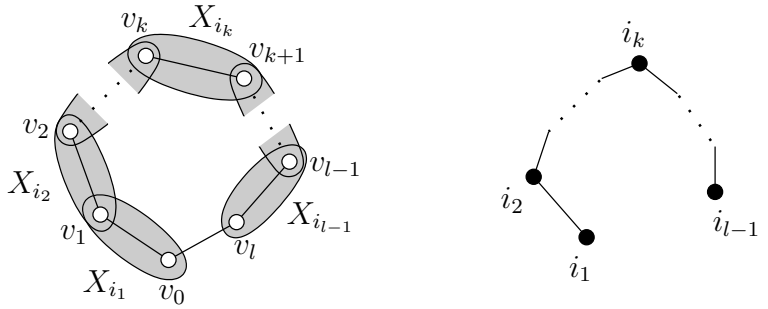
Prova.
($\Rightarrow$) Seja G uma floresta e $C_1, \ldots, C_l$ suas componentes. Seja C^* a componente de G com maior largura em árvore, ou seja, pela Propriedade 3, $LA(C^*) = LA(G)$. Considere a seguinte decomposição em árvore de C^*:

1. Para cada aresta $e_k = (u, v)$ de C^*, seja $X_{i_k} = \{u, v\}$;
2. Para cada vértice $v_k \in V(G)$, seja $X_{j_k} = \{v_k\}$;
3. Seja $T = (I, F)$, $I_i = \bigcup_{k=1}^{|E(G)|} i_k$, $I_j = \bigcup_{k=1}^{|V(G)|} j_k$, $I = I_i \cup I_j$ e $F = \{(i, j) : X_i \cap X_j \neq \emptyset, i \in I_i, j \in I_j\}$. Basta ver agora que T é isomorfo a um grafo obtido pela subdivisão de arestas de C^*, e logo T contém um ciclo se e somente se C^* também contém. Portanto, como C^* é uma componente de uma floresta, T é uma *árvore*.

Pode-se ver então que $(\mathcal{X} = \{X_i : i \in I\}, T = (I, F))$ é uma decomposição em árvore de C^*. É fácil ver que $LA(C^*) = 1$, já que qualquer X_{i_k} tem tamanho máximo e $|X_{i_k}| = 2$. Logo, pela Propriedade 3 e pela maximalidade de C^*, $LA(G) = 1$.

($\Leftarrow$) Seja G um grafo com largura em árvore unitária. Suponha, por absurdo, que G contém um ciclo C. Como $LA(G) = 1$, as partes de G têm no máximo dois vértices, e considere ainda, sem perda de generalidade, uma decomposição em árvore $D = (\mathcal{X}, T = (I, F))$ com *todas* as partes contendo de fato dois vértices; isto equivale a dizer que D é uma decomposição boa. A idéia é mostrar que não é possível construir uma decomposição em árvore de G usando tais partes, ou seja, D não existe.

Considere então um ciclo $C = \langle v_0, e_1, v_1, e_2, \ldots, e_{l-1}, v_{l-1}, e_l, v_0 \rangle$ de G e as partes $X_{i_k} = \{u, v : e_k = (u, v)\}$, como ilustrado na Figura 3.7.a.



(a) (b)

Figura 3.7. Tentativa de decomposição em árvore de um ciclo.

Como X_{i_1} e X_{i_2} têm como interseção v_1, e v_1 não pertence a nenhuma outra parte, não pode haver nenhum nó no caminho entre i_1 e i_2 — do contrário, a terceira condição da definição de decomposição em árvore seria violada — e como i_1 e i_2 devem ser conectados, logo (i_1, i_2) deve pertencer a F. Da mesma forma, como $X_{i_k} \cap X_{i_{k+1}} = \{v_k\}$, e nenhuma outra parte de I contém v_k, então $(i_k, i_{k+1}) \in F$. Até a parte i_l, T é um *caminho*, ou seja, $(i_k, i_{k+1}) \in F, 1 \leq k < l$, como mostra a Figura 3.7.b.

No entanto, $X_{i_1} \cap X_{i_l} = \{v_0\}$, e como $v_0 \notin \bigcup_{k=2}^{l-1} X_{i_k}$, não pode haver nós entre i_1 e i_l, e logo T não pode ser uma árvore. Mas isto é um absurdo, já que D é uma decomposição em árvore por hipótese, e logo G não contém ciclos, sendo portanto uma floresta. □

Como a classe $\mathcal{F}$ dos grafos que são florestas é fechada com relação a tomada de menores — todo menor de uma floresta é ainda uma floresta — então pode-se caracterizar $\mathcal{F}$ por um conjunto de menores proibidos, como permitido pela Proposição 2.8.

Lema 3.4. *A classe dos grafos com largura em árvore unitária (no máximo 1) é exatamente a classe de grafos que tem K_3 como menor proibido, Proib$_{\preccurlyeq}(\{K_3\})$.*

Prova. Pelo Teorema 3.3, e pela definição de classe de menores proibidos, basta mostrar que um grafo G contém um ciclo se e somente se G tem K_3 como menor. A condição necessária é trivial: basta tomar dois vértices adjacentes de um ciclo C qualquer de G, com $|C| > 3$, e contrair recursivamente as arestas não incidentes a estes dois vértices; o resultado final será um K_3.

Para a prova da condição suficiente, suponha, sem perda de generalidade, que $G = MK_3$, e logo existem subconjuntos U_1, U_2 e U_3 de vértices, cada um induzindo um subgrafo conexo em G, tais que $G/U_1/U_2/U_3 = K_3$. Considere então três pares de vértices $u_{i,j}$, $1 \leq i, j \leq 3$, $i \neq j$, tais que existe uma aresta entre $u_{i,j}$ e $u_{j,i}$, $e_{i,j}$, onde $u_{i,j}$ pertence a U_i e $u_{j,i}$ pertence a U_j. Como cada U_i induz um subgrafo conexo em G, seja P_i o caminho unindo cada par de vértices acima em U_i. A Figura 3.8 mostra G e cada elemento acima.

Basta agora ver que $\langle u_{1,3}, P_1, u_{1,2}, e_{1,2}, u_{2,1}, P_2, u_{2,3}, e_{2,3}, u_{3,2}, P_3, u_{3,1}, e_{1,3}, u_{1,3}\rangle$ induz um ciclo em G, o que conclui a prova. □

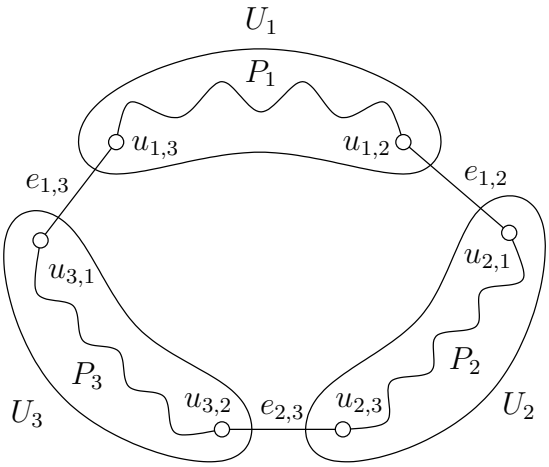


Figura 3.8. Um grafo G possui um ciclo se e somente se tem K_3 como menor.

3.4 DEA para ciclos

Pelo Teorema 3.3, viu-se que a presença de um ciclo em um grafo G inviabiliza a largura em árvore unitária para G. No entanto, apenas a presença de um ciclo não torna a largura em árvore de G grande, como se pode deduzir do próximo lema.

Lema 3.5. *Todo ciclo tem largura em árvore 2.*

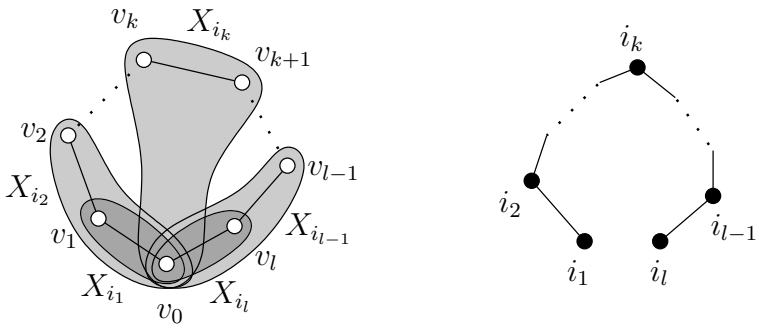
Prova. Pelo Teorema 3.3, e por conter ao menos uma aresta, um ciclo deve ter largura em árvore no mínimo 2. Logo, para provar o lema, basta encontrar uma decomposição em árvore de largura 2 para qualquer ciclo, já que esta certamente será mínima. Na prova da suficiência do Teorema 3.3 utilizou-se uma construção de modo a demonstrar a impossibilidade de se encontrar uma decomposição em árvore de largura unitária para um ciclo. Tal construção pode ser aproveitada para se obter uma decomposição semelhante, mas de largura um pouco maior.

Considere então um ciclo $C = \langle v_0, e_1, v_1, e_2, \ldots, e_{l-1}, v_{l-1}, e_l, v_0 \rangle$ e as partes $X_{i_k} = \{u, v \,:\, e_k = (u, v)\}$. A dificuldade está no fato de que $v_0 \in X_{i_1} \cap X_{i_l}$, mas não está contido em nenhuma outra parte no caminho entre X_{i_1} e X_{i_l}. A resolução é então simples, já que não há mais obrigação em manter a largura unitária: adicionar v_0 a cada parte X_{i_j}, $1 < j < l$. Logo,

$$D = \left(\left\{ X'_{i_k} = \{u, v \,:\, (u, v) = e_k\} \cup \{v_0\} \right\}_{1 \leq k \leq l+1}, T = \langle i_1, i_2, \ldots, i_l \rangle \right)$$

é uma decomposição em árvore de largura 2 de C, como mostra a Figura 3.9. Vale notar que T é um caminho, neste caso. □

A partir da decomposição em árvore D encontrada na prova do Lema 3.5, pode-se perceber que D é também uma decomposição em árvore para $G = (V(C), E(C) \cup \{(v_0, v_i \,:\, 1 < i < l\})$, já que toda aresta adicionada a C em G está coberta por alguma parte de D. De fato, a possibilidade de adicionar arestas a um grafo sem aumentar a



(a) (b)

Figura 3.9. Decomposição em árvore de um ciclo.

sua largura em árvore — já que a mesma decomposição em árvore é usada — é uma propriedade bastante útil, e merece uma descrição formal como nova ferramenta.

Propriedade 5. Se $D = (\mathcal{X}, T)$ é uma decomposição em árvore de $G = (V, E)$, e $\exists i \in I : v, w \in X_i$, $v \neq w$, então D é também uma decomposição em árvore de $G + (v, w) = (V, E \cup \{(v, w)\})$, e vice-versa.

Prova. O lema é trivial. Basta ver que a aresta acrescentada a G é coberta pela parte em D contendo suas extremidades; logo, D é também uma decomposição em árvore do novo grafo. O sentido contrário também vale e é imediato pelo simples fato de que D é ainda uma decomposição em árvore de qualquer subgrafo de G obtido por retirada de arestas (embora não seja necessariamente a decomposição de menor largura). □

Considere agora um C_4 com conjunto de vértices $\{v_1, v_2, v_3, v_4\}$, e logo, usando o Lema 3.5, pode-se ver que C_4 tem largura em árvore 2 e que $D = (\{\{v_1, v_2, v_3\}, \{v_3, v_4, v_1\}\}, T = (\{i_1, i_2\}, \{(i_1, i_2)\}))$ é uma decomposição em árvore mínima de C_4. Logo, D é também uma DEA de $C' = C + (v_1, v_3)$, pela Propriedade 5. Suponha agora que se queira adicionar a aresta (v_2, v_4) a C'. Após algumas tentativas exaustivas, conclui-se que não se pode mais encontrar uma decomposição em árvore D de largura 2 para $C'' = C' + (v_2, v_4)$, pois pelo menos uma parte de uma DEA para C'' deve conter todos os vértices de C. Pode-se ver que $C'' = K_4$, e a próxima seção mostra que os dois últimos fatos não são coincidência.

3.5 DEA para cliques

A Propriedade 1 estabeleceu uma relação entre a conectividade de um grafo G e uma árvore T, onde T é definida por uma decomposição em árvore $D = (\mathcal{X}, T)$ de G. A próxima propriedade abaixo estabelece uma relação semelhante para um subconjunto de vértices de G.

Propriedade 6. Sejam $D = (\mathcal{X} = \{X_i : i \in I\}, T = (I, F))$ uma decomposição em árvore de $G = (V, E)$, e $W \subseteq V$. Então ou existe um $i \in I$ tal que $W \subseteq X_i$ ou existem vértices $w_i, w_j \in W$ tais que $T_{w_i} \cap T_{w_j} = \emptyset$.

Prova. Seja $W = \{w_1, w_2, \ldots, w_r\}$, e $\mathcal{T}_W = \{T_{w_i}\}_{1 \leq i \leq r}$ a família de sub-árvores induzidas por cada elemento de W. Se não existem vértices $w_i, w_j \in W$ tais que $T_{w_i} \cap T_{w_j} = \emptyset$ então $\mathcal{T}_W$ satisfaz a propriedade Helly pelo Lema 2.2, e logo $\bigcap \mathcal{T}_W = T_{w^*}$, e existe um X^* tal que $W \subseteq X^* \in T_{w^*}$. □

Considere agora o problema de encontrar uma decomposição em árvore mínima para uma clique. Pelo Lema 3.5, sabe-se que $LA(K_3) = 2$, justamente pela dificuldade encontrada na prova do Teorema 3.3: sendo v_1, v_2 e v_3 os vértices de K_3, não se pode construir uma boa decomposição em árvore com $X_1 = \{v_2, v_3\}$, $X_2 = \{v_1, v_3\}$, e $X_3 = \{v_1, v_2\}$, pois $X_1 \not\supseteq X_2 \cap X_3$, $X_2 \not\supseteq X_1 \cap X_3$ e $X_3 \not\supseteq X_1 \cap X_2$, impossibilitando a satisfação de [P3]. E logo, uma decomposição em árvore mínima de K_3 deve ter uma parte contendo todos os seus vértices.

Na verdade, essa dificuldade é encontrada na tentativa da construção de uma decomposição em árvore de largura $k-1$ para qualquer clique de tamanho $k+1$. De fato, pode-se mostrar que isso é impossível, e logo toda clique W deve estar contida em uma parte de uma decomposição, forçando a largura em árvore do grafo (e da clique) para, no mínimo, $|W| - 1$.

Suponha que se queira contornar essa dificuldade e encontrar uma DEA de largura no máximo $k-1$ para clique de tamanho $k+1$, e logo suponha que exista uma DEA $D = (\mathcal{X}, T = (I, F))$ de uma clique W com $V(W) = \{v_0, v_1, \ldots, v_k\}$ tal que $LA(D) \leq |W| - 2$. Seja $i \in I$ um nó de T tal que $W - v_0 \subseteq X_i$. Sejam ainda $j, j' \in T$ tais que $\{v_0, v_1\} \subseteq X_j$ e $\{v_0, v_2\} \subseteq X_{j'}$. Como os três conjuntos X_i, X_j e $X_{j'}$ se interceptam, é necessário então, pela restrição [P3], que tenham uma interseção em comum, e logo ou $v_2 \in X_j$ ou $v_1 \in X_{j'}$. Isso é fácil de perceber porque $W[\{v_0, v_1, v_2\}] = K_3$. Suponha, sem perda de generalidade, que $v_2 \in X_j$, e seja agora j' um nó tal que $\{v_0, v_3\} \in X_{j'}$. Pelo mesmo motivo anterior, ou seja, pela necessidade da existência de uma interseção comum entre X_i, X_j e $X_{j'}$ dada a interseção não nula entre estes, dois a dois, assume-se, s.p.d.g., que $v_3 \in X_j$. Continuando desta forma, chega-se a $X_j \supseteq \{v_0, v_1, \ldots, v_{k-1}\}$ e, como existe uma aresta (v_0, v_k), a existência de um nó $j' \in T$ tal que $\{v_0, v_k\} \subseteq X_{j'}$. O mesmo argumento da necessidade de interseção comum vale, e logo conclui-se que $W \subseteq X_j$.

Conclui-se então que toda clique de um grafo G deve estar contida em alguma parte de qualquer decomposição em árvore de G. A propriedade seguinte formaliza tal resultado, acrescentando ainda outro bastante relacionado.

Propriedade 7. Seja $D = (\mathcal{X} = \{X_i : i \in I\}, T = (I, F))$ uma decomposição em árvore de $G = (V, E)$.

1. (*Continência de clique*). Se $W \subseteq V$ forma uma clique em G, então $\exists i \in I : W \subseteq X_i$.
2. (*Continência de subgrafo bipartite completo*). Se A, $B \subseteq V$ são tais que cada vértice em A é adjacente a cada vértice em B, então $\exists i \in I : A \subseteq X_i$ ou $B \subseteq X_i$.

Prova.

1. Se $|W| = 1$ ou $|W| = 2$ então a propriedade claramente vale como conseqüência da definição de decomposição em árvore, pelas restrições [P1] e [P2], respectivamente.

Suponha então inicialmente $W = K_3$, e seja $v \in W$. Por hipótese de indução, seja $i \in T$ tal que $W - v \subseteq X_i$. Seja agora $T_v = (I', F')$ a sub-árvore de T cujas partes contém v. Suponha $i \notin I'$, caso contrário $W \subseteq X_i$. Seja j o nó em T_v mais próximo de i em T. Tome um vértice qualquer w de $W - v$, e seja $j' \in I'$ tal que $\{v, w\} \in X_{j'}$; tal parte deve existir, já que $(v, w) \in W$. Mas $X_i \cap X_{j'} \supseteq \{v, w\}$, e, como j está no caminho entre i e j' em T, por [P3] deve-se ter $\{v, w\} \subseteq X_i \cap X_{j'} \subseteq X_j$. Como X_j contém qualquer vértice $w \in W - v$ e v, segue que $W \subseteq X_j$.
Uma prova mais curta pode ser obtida diretamente com base na propriedade Helly. Seja $\mathcal{T}_W = \{T_{w_i} = \{i \in I : w_i \in X_i\}\}_{1 \leq i \leq r}$. Basta perceber que, como sempre existe uma aresta entre quaisquer dois vértices u e v de W, então $T_u \cap T_v \neq \emptyset$. Pela Propriedade 6, tem-se $W \subseteq X^*$, $X^* \in \bigcap \mathcal{T}_W$.

2. Sejam G, D, A e B como no enunciado, e suponha $B \not\subseteq X_i$ para todo $i \in I$. Sejam $a_i, a_j \in A$, e $b_i, b_j \in B$ tais que T_{b_i} e T_{b_j} sejam disjuntos em vértices. A existência de b_i e b_j é garantida pela Propriedade 6. Pela existência da aresta (a_i, b_i), deve-se ter $t_i \in T_{b_i}$ tal que $a_i \in X_{t_i}$, assim como, devido a (a_i, b_j), deve existir $t_j \in T_{b_j}$ tal que $a_i \in X_{t_j}$. Por [P3], todos os vértices no caminho entre t_i e t_j em T contém a_i, e logo, seja $k \in t_i T t_j$, e portanto $a_i \in X_k$. Suponha ainda, sem perda de generalidade, que t_i e t_j são extremidades do caminho mínimo (único) em T unindo T_{b_i} e T_{b_j}, como mostra a Figura 3.10.

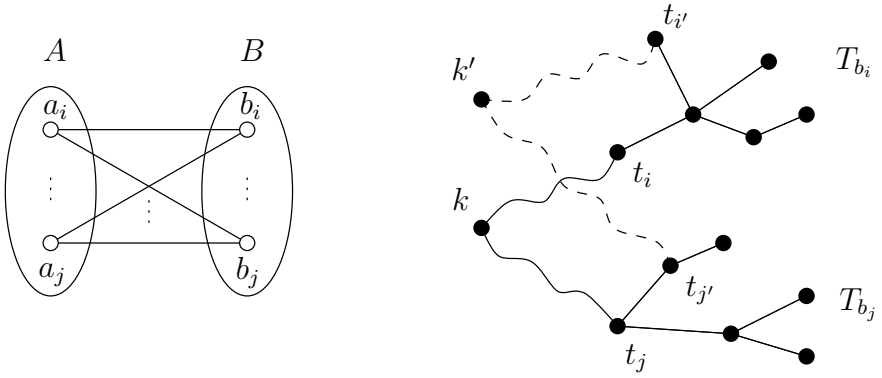


Figura 3.10. Se $A, B \subseteq V$ induzem um subgrafo bipartite completo em $G = (V, E)$, então ou A ou B estão contidos em alguma parte de uma DEA de G.

Seja agora $a_j \in A$, e, de modo análogo a a_i, sejam $t_{i'} \in T_{b_i}$ e $t_{j'} \in T_{b_j}$ tais que $a_j \in X_{t_{i'}}$ e $a_j \in X_{t_{j'}}$, e $k' \in t_{i'} T t_{j'}$. Mas $k' T t_{i'} T_{b_i} t_i T k T t_j T_{b_j} t_{j'} T k'$ formam um ciclo a menos que $t_{i'} = t_i$ e $t_{j'} = t_j$. Logo, $a_j \in X_k$, e como todo vértice em A deve pertencer a X_k, segue que $A \subseteq X_k$.

□

A partir da Propriedade 7, uma outra propriedade, semelhante à Propriedade 5, pode ser encontrada:

Propriedade 8. Sejam v e w vértices não adjacentes em $G = (V, E)$ com pelo menos $k + 1$ vizinhos em comum. Se a largura em árvore de G é no máximo k, então a largura

em árvore de $G + (v, w)$ é também no máximo k. Além disso, qualquer decomposição em árvore de G com largura no máximo k é também uma decomposição em árvore de $G + (v, w)$ com largura no máximo k, e vice-versa.

Prova. Seja $D = (\mathcal{X}, T)$ uma decomposição em árvore de G com largura no máximo k. Pela Propriedade 7, ou existe um $i \in I$ com $v, w \in X_i$ — e logo a propriedade segue diretamente pela Propriedade 5 — ou existe um $i \in I$ com X_i contendo o conjunto W de todos os vizinhos em comum de v e w. Neste caso, poderiam ser adicionadas arestas unindo todos os pares de vértices não adjacentes em W, resultando em pelo menos duas cliques com $k + 2$ vértices, $W + v$ e $W + w$. Novamente pela Propriedade 5, essa nova decomposição em árvore D' deve ter a mesma largura de D, um absurdo já que nenhuma decomposição em árvore com largura no máximo k pode conter uma clique com $k + 2$ vértices. Logo, por contradição, o primeiro caso vale, e a propriedade segue. □

Com a ajuda da Propriedade 7, tem-se mais ferramentas para encontrar uma caracterização de uma nova classe de grafos de largura pequena e limitada.

Teorema 3.6. *A classe dos grafos com largura em árvore no máximo 2 é exatamente a classe de grafos que tem K_4 como menor proibido, Proib$_{\preccurlyeq}(\{K_4\})$.*

Prova. O teorema equivale a dizer que um grafo G tem largura em árvore no máximo 2 se e somente se G não possui K_4 como menor. A condição necessária é imediata, já que se G tem largura em árvore no máximo 2 então a maior clique de G tem tamanho 3, e uma clique de tamanho 4 resultaria numa largura no mínimo 3 para G.

Suponha então, s.p.d.g., que $G = MK_4$ e, no pior caso, é maximal em arestas. Pela Proposição 2.7, se $G = MK_4$ então $G = TK_4$, já que $\Delta(K_4) = 3$, e logo, G não possuir K_4 como menor é equivalente a G não possuir uma subdivisão de K_4 como subgrafo. A idéia da prova é mostrar, por indução, que G pode ser montado por colagem de K_2 a partir de triângulos, e logo, claramente $LA(G) \leq 2$, já que desta forma a maior clique de G é um K_3.

Se $G = K_3$, tem-se a base da indução, e suponha então $|G| \geq 4$ como passo. Como G não é completo, sejam S um separador em G tal que $|S| = \kappa(G)$, C_1 e C_2 componentes de $G - S$, e $G_1 = G[C_1 \cup S]$ e $G_2 = G[C_2 \cup S]$. Como S é um separador minimal, todo vértice em S possui um vizinho em C_1 e outro em C_2. Se $|S| \geq 3$, então G possui pelo menos 3 caminhos disjuntos em vértices entre $v_1 \in C_1$ e $v_2 \in C_2$, P_1, P_2 e P_3. Como $\kappa(G) \geq 3$, existe um outro caminho P entre u e v em $G \setminus \{v_1, v_2\}$ tais que u e v pertencem a caminhos diferentes entre P_1, P_2 e P_3 e possam, desta forma, ter pelo menos 3 caminhos independentes entre si. Na Figura 3.11, $u \in P_1$ e $v \in P_2$, e logo P é um terceiro caminho unindo u a v.

Mas $P \cup P_1 \cup P_2 \cup P_3 = TK_4$, um absurdo, e logo $\kappa(G) \leq 2$. Logo, $|S| \leq 2$ e, pela maximalidade de G, $G[S] = K_2$. Como $G_1 \cap G_2 = S$ e $G_1 \cup G_2 = G$, G pode ser obtido a partir de G_1 e G_2 por colagem ao longo de $S = K_2$, o resultado segue por indução. □

Como ilustrado pelos lemas sobre caracterização de grafos com largura em árvore limitada, uma boa abordagem é considerar sub-estruturas proibidas em um determinado grafo e então classificá-lo. Infelizmente, esta abordagem pode não ser eficiente a medida

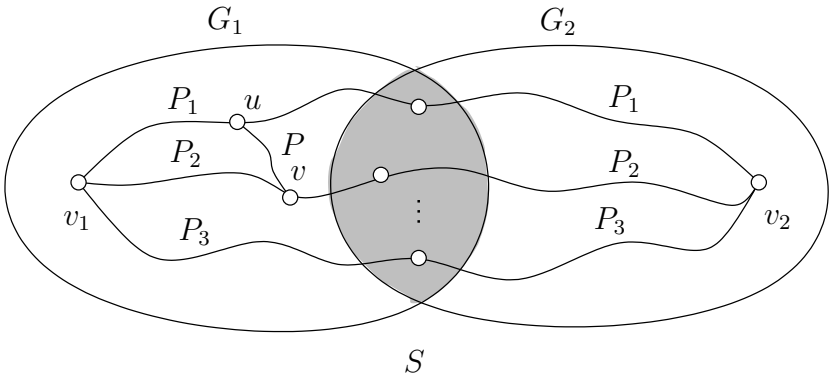


Figura 3.11. Um grafo G não tem K_4 como menor se e somente se tem largura em árvore no máximo 2.

em que se aumenta a largura máxima permitida para uma decomposição em árvore, já que os conjuntos de obstrução também crescem.

Uma idéia atraente é então procurar definir algoritmos de reconhecimento de classes dos grafos com largura em árvore limitada de um modo geral, que não precisem de uma definição explícita dos menores proibidos de cada classe, para cada largura. Tais algoritmos serão discutidos adiante, no próximo capítulo; no entanto, ao longo deste capítulo, novos conceitos serão apresentados e discutidos de modo a amadurecer teoricamente uma abordagem algorítmica.

3.6 DEA para grafos cordais

Prosseguindo com a busca de caracterizações para grafos de largura em árvore limitada, pode-se ver que a largura em árvore de um grafo G é claramente limitada inferiormente pelo tamanho da maior clique em G, ou seja, $LA(G) \geq \omega(G) - 1$ como conseqüência da Propriedade 7. Uma classe de grafos interessante é então a classe de grafos cordais, pelo seguinte resultado:

Teorema 3.7. *Um grafo $G = (V, E)$ é cordal se e somente se admite uma decomposição em árvore $D = (\mathcal{X}, T = (I, F))$ onde $\mathcal{X}$ corresponde à família de cliques maximais de G.*

Prova.
($\Rightarrow$) Suponha que G seja cordal. A prova da condição necessária é feita por indução no tamanho de G. Assuma, portanto, que a condição é válida para todos os grafos com menos vértices que G. Como base da indução, se G é completo então T é composta de somente um nó; se G é desconexo com componentes $C_1, \ldots, C_r$, então, pela hipótese de indução, existe uma árvore T_i para cada componente C_i, e logo, pela adição de um nó i^* à união das (sub-)árvores T_i, tal que X_{i^*} seja igual a alguma clique maximal de alguma componente de G, pode-se obter uma árvore T para G satisfazendo à condição (de forma semelhante à construção abordada na prova da Propriedade 3).

Assuma portanto, que G é conexo e não completo, e seja w um vértice simplicial de G. Defina agora $W = \{w\} \cup N_G(w)$, claramente uma clique maximal de G. Sejam ainda $U = \{u \in W : N_G(u) \subset W\}$ e $Y = W \setminus U$. Considere agora o grafo $G' = G[V \setminus U]$, que

é também cordal, e, como tem menos vértices que G e pela hipótese de indução, admite uma decomposição em árvore $D' = (\mathcal{X}', T' = (I', F'))$ satisfazendo o teorema.

Seja então K a clique maximal de G' contendo Y. Para construção de uma decomposição $D = (\mathcal{X}, T = (I, F))$ de G a partir de D' e G', devem-se considerar dois casos:

1. $Y = K$**.** Neste caso, seja $i \in I'$ tal que $X_i = K$. Logo, basta substituir X_i por W em D':

$$D = (\mathcal{X} = \{X_j : X_j \in \mathcal{X}', j \neq i\} \cup \{X_i = W\}, T = (I', F'))$$

2. $Y \subset K$**.** Como no caso anterior, seja novamente $i \in I'$ tal que $X_i = K$. A nova decomposição é obtida acrescentando-se um novo vértice i' adjacente a i tal que $X_{i'} = W$:

$$D = (\mathcal{X} = \mathcal{X}' \cup \{X_{i'} = W\}, T = (I' \cup \{i'\}, F' \cup \{(i, i')\}))$$

($\Leftarrow$) Seja $G = (V, E)$ um grafo com uma decomposição em árvore $D = (\mathcal{X}, T = (I, F))$ como no enunciado do teorema. Inicialmente, vale notar que se $(u, v) \in E$, então as sub-árvores T_u e T_v se interceptam, já que a clique maximal que contém u e v deve estar em T_u e T_v.

Suponha agora, por absurdo, que G contenha um ciclo sem cordas $C = \langle v_0, v_1, \ldots, v_{k-1}, v_0 \rangle$, com $k > 3$, e com sub-árvores correspondentes $T_0, T_1, \ldots, T_{k-1}$ em T. Seja $\mathcal{P} = \{P_i\}_{0 \leq i \leq k-1}$ uma família de caminhos em T tal que cada $P_i \subseteq T_i$, $P_i \cap P_{(i-1) \mod k} \neq \emptyset$ e $P_i \cap P_{(i+1) \mod k} \neq \emptyset$. Como $T_i \cap T_{(i-1) \mod k} \neq \emptyset$ e $T_i \cap T_{(i+1) \mod k} \neq \emptyset$, pelas arestas em C, tais caminhos sempre existem. Vale notar que $P_{(i-1) \mod k} \cap P_i \cap P_{(i+1) \mod k} = \emptyset$, para $0 \leq i \leq k-1$, já que, caso contrário, a interseção entre os caminhos implicaria numa clique contendo $v_{(i-1) \mod k}$, v_i e $v_{(i+1) \mod k}$, o que representa uma corda em C, violando a hipótese de absurdo. Além disso, pelo Lema 2.3, $\mathcal{P}$ deve satisfazer a propriedade Helly, e logo, para todo i, $P_{(i-1) \mod k} \cap P_{(i+1) \mod k} = \emptyset$. Basta agora notar que a união dos P_i representa um ciclo (não induzido) em T, um absurdo, já que T é uma árvore. □

A Figura 3.12 ilustra uma decomposição em árvore mínima $D = (\mathcal{X}, T)$ para um grafo cordal G onde todas as partes de D correspondem a cliques maximais em G; em (a) pode-se ver o grafo G, em (b) a árvore T associada a D e G, em (c) as partes de D e em (d) pode-se perceber melhor as cliques maximais de G.

Logo, a decomposição em árvore de um grafo cordal revela propriedades desejáveis, de modo que pode-se determinar prontamente a largura em árvore de um grafo cordal.

Corolário 3.8. *Se G é um grafo cordal, então $LA(G) = \omega(G) - 1$.*

Prova. O corolário segue diretamente do Teorema 3.7 e da Propriedade 7: basta ver que qualquer decomposição em árvore mínima de um grafo G cordal tem apenas cliques em suas partes, e logo a largura em árvore é realizada nas partes que contém as maiores

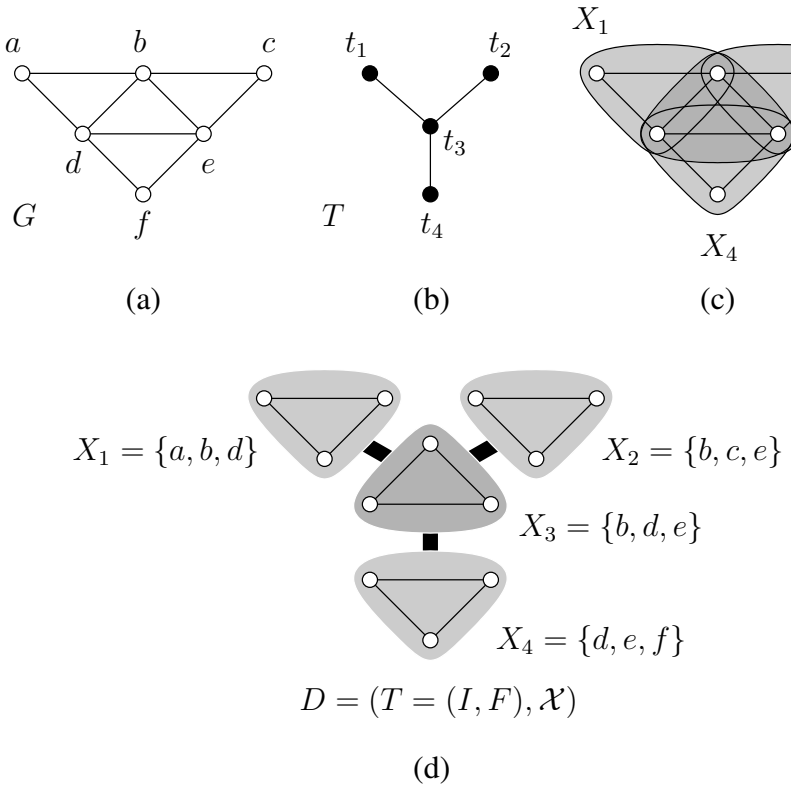


Figura 3.12. Exemplo de decomposição em árvore para um grafo cordal.

cliques. Portanto, pela definição de largura em árvore, $LA(G) = \omega(G) - 1$, o maior tamanho de clique de G menos 1. □

Como encontrar uma decomposição em árvore para um grafo cordal H depende da identificação de todas as cliques maximais de H, o que pode ser feito em tempo linear usando um esquema de eliminação perfeito para H, tem-se um limitante superior para a largura em árvore de todos os subgrafos G de H.

Lema 3.9. *Sejam G um grafo e H um grafo cordal obtido a partir de uma triangulação qualquer de G. Então $LA(G) \leq \omega(H) - 1$.*

Prova. Como G é um subgrafo de H, segue que $LA(G) \leq LA(H)$ pela Propriedade 2. Mas pelo Lema 3.9 anterior, $LA(H) = \omega(G) - 1$, o que conclui a prova. □

Um resultado ainda mais importante pode ser obtido se, para um grafo G, obtém-se uma triangulação de G onde cada aresta da triangulação satisfaz a Propriedade 5.

Lema 3.10. *Para todo grafo $G = (V, E)$, existe uma triangulação E' de G resultando no grafo $H = (V, E \cup E')$ tal que $LA(G) = LA(H)$.*

Prova. Seja $D = (\mathcal{X}, T = (I, F))$ uma decomposição em árvore mínima de G. Considere o conjunto E' de arestas definido da seguinte forma: para cada par u, v de vértices em G tais que $(u, v) \notin E$ e $u \in X_i$, $v \in X_i$, para algum $i \in I$, acrescente (u, v) a E'. Claramente D é também uma decomposição em árvore de $H = (V, E \cup E')$. Além disso,

como cada parte de D em H corresponde a uma clique maximal, pelo Lema 3.7, H é cordal, e logo E' é a triangulação procurada. Pela minimalidade de D, segue também que $LA(H) = LA(G)$. □

Uma conseqüência direta do Lema 3.10 é a equivalência entre os problemas (de otimização) de encontrar uma decomposição em árvore de largura mínima para um grafo G e o de encontrar uma cordalização de G que tenha uma clique de tamanho mínimo dentre todas as cordalizações possíveis de G. Tal equivalência é aqui posta explicitamente para uso posterior.

Corolário 3.11. *Encontrar uma decomposição em árvore mínima de um grafo G é equivalente a encontrar uma cordalização H de G que minimiza $\omega(H)$.*

Prova. A prova é direta, conseqüência dos Lemas 3.9 e 3.10. □

Outro resultado importante do Lema 3.10 é a possibilidade de relacionar um grafo G com largura em árvore no máximo k a uma k-cordalização de G. Para tanto, alguns outros conceitos se fazem necessários.

Definição 14 (Aresta necessária). Seja $G = (V, E)$ um grafo.

1. Um par (u, v) é uma *aresta necessária para largura em árvore* k de G se, para toda decomposição em árvore $D = (\mathcal{X} = \{X_i : i \in I\}, T = (I, F))$ com largura no máximo k de G, existe um j tal que $u, v \in X_j$.
2. Um par (u, v) é uma *aresta necessária para k-cordalização de G* se, para toda k-cordalização $H = (V, E')$ de G, $(u, v) \in E'$.

Na verdade, os conceitos de aresta necessária estão intimamente relacionados:

Lema 3.12. *Sejam $G = (V, E)$ um grafo de largura em árvore no máximo k e $u, v \in V$. As duas assertivas abaixo são equivalentes:*

1. *(u, v) é uma aresta necessária para largura em árvore k em G.*
2. *Toda k-cordalização de G contém a aresta (u, v).*

Prova. Para provar o lema, basta mostrar a relação direta entre uma decomposição em árvore de um grafo G e uma cordalização de G. Seja então $D = (\mathcal{X} = \{X_i\}_{i \in I}, T = (I, F))$ uma decomposição em árvore arbitrária de G. Seja $E' = \{(u, v) : u \in X_i, v \in X_i, X_i \in \mathcal{X}, (u, v) \in E(G)\}$. Como a adição de E' a G resulta num novo grafo G' onde D é ainda uma decomposição em árvore (pela Propriedade 5). Como cada X_i induz uma clique maximal em G', segue pelo Teorema 3.7 que G' é cordal, e logo E' é uma triangulação. Além disso, se $LA(D) \leq k$, então G' é uma k-cordalização de G. O sentido converso é também válido: se G' é uma k-cordalização de G, então admite uma decomposição em árvore de largura no máximo k, que é também válida — e de mesma largura — para G pela Propriedade 2. Logo, se (u, v) é uma aresta necessária para largura em árvore k de G, então u e v pertencem a alguma parte de qualquer decomposição em árvore D de G, $LA(D) \leq k$, e pertencem também a qualquer k-cordalização de G. O mesmo vale se (u, v) pertence a toda k-cordalização G' de G, já que, desta forma, u e v devem pertencer a mesma parte da decomposição em árvore de G obtida através de G'. □

A utilidade de uma aresta necessária (u, v) é evidente: se G é um grafo com largura em árvore no máximo k, então a adição de (u, v) a G não aumenta sua largura em árvore. Desta forma, a adição de arestas necessárias para largura em árvore k em um grafo G resulta numa cordalização de G satisfazendo o Lema 3.10, uma k-cordalização. O lema seguinte introduz uma condição suficiente para a existência de uma aresta necessária:

Lema 3.13. *Sejam $G = (V, E)$ um grafo e $u, v \in V$. Se existem $k+1$ caminhos disjuntos em vértices entre u e v em G, então a aresta (u, v) é necessária para largura em árvore k em G.*

Prova. Se $(u, v) \in E$ então, claramente (u, v) é necessária. Considere então que $(u, v) \notin E$. Seja $D = (\mathcal{X} = \{X_i : i \in I\}, T = (I, F))$ uma decomposição em árvore de G com largura no máximo k. Suponha que existam $k + 1$ caminhos disjuntos em vértices entre u e v, $P_1, \ldots, P_{i+1}$, e identifique, para cada P_i, todos os vértices internos, tanto em G como em D, em um vértice v_i, $1 \leq i \leq k+1$. A decomposição em árvore $D' = (\mathcal{X}', T')$ resultante é uma decomposição do grafo resultante G' com largura também no máximo k, pela Propriedade 4. A necessariedade de (u, v) segue então do fato de que u e v têm $k + 1$ vizinhos em G' e pela Propriedade 8. □

O Lema 3.13 resulta numa importante ferramenta para encontrar uma decomposição em árvore de um grafo. Se um par de vértices é separado por $k + 1$ caminhos disjuntos, então dois casos podem ocorrer: ou a aresta unindo o par é necessária e o grafo tem largura em árvore no máximo k, ou os vértices internos a todos os $k + 1$ caminhos estão em uma mesma parte e a largura em árvore do grafo é estritamente maior que k.

Como Bodlaender (2000) aponta, é também possível prover ainda uma condição necessária e suficiente para a existência de uma aresta necessária:

Lema 3.14. *Seja $G = (V, E)$ um grafo com largura em árvore no máximo k, e sejam ainda u e v vértices não adjacentes em G. Então (u, v) é uma aresta necessária se e somente se não existe um subconjunto $S \subseteq V$ de vértices com $|S| \leq k+1$, $\{u, v\} \cap S = \emptyset$, tal que S separa u e v em G e, para toda componente C de $G-S$, o grafo $G[C \cup S]+K(S)$ tem largura em árvore k.*

Prova.
($\Leftarrow$) Sejam S um conjunto de vértices satisfazendo as condições do enunciado, e $C_1, \ldots, C_r$ as componentes de $G - S$. Sejam agora $D_1, \ldots, D_r$ as decomposições em árvore de largura no máximo k das respectivas componentes. De forma semelhante a construção ilustrada na prova da Propriedade 3, seja D a decomposição em árvore de G obtida pela união disjunta das D_i, $1 \leq i \leq r$, e da adição de um novo nó i^* tal que $X_{i^*} = S$, que é feito adjacente aos nós i_j de cada D_j tais que $X_{i_j} \supseteq S$. Como cada decomposição tem largura no máximo k e $|S| \leq k + 1$, segue que $LA_G(D) \leq k$. Além disso, não existe em D uma parte X^* tal que $u, v \in X^*$, e logo (u, v) não é necessária. Como Bodlaender observa, essa construção, por sua vez, é semelhante a usada por Arnborg, Corneil e Proskurowski para reconhecimento de grafos com largura em árvore k.

($\Rightarrow$) Suponha agora que (u, v) não é necessária, e seja $D = (\mathcal{X} = \{X_i\}_{i \in I}, T = (I, F))$ uma decomposição em árvore de G com largura no máximo k, onde, por hipótese, u e v pertencem a partes distintas. Sejam ainda T_u e T_v as sub-árvores induzidas em T por u e v, respectivamente, e sejam $i_u \in T_u$ e $i_v \in T_v$ os nós que realizam o caminho P de menor distância entre algum nó de T_u e algum nó de T_v em T. Se i_u e i_v são adjacentes em T, então seja D' a decomposição em árvore obtida a partir de D pela subdivisão de (i_u, i_v) com o acréscimo de um novo nó i^* tal que $X_{i^*} = X_{i_u} \cap X_{i_v}$. Se $|P| > 1$, então seja i^* um nó arbitrário em $i_u P i_v$. Em ambos os casos, pode-se ver que $S = X_{i^*}$ é tal que $\{u, v\} \cap S = \emptyset$, S é um uv-separador, $|S| \leq k + 1$ e, para qualquer componente C de $G - S$, $G[C \cup S] + K(S)$ é um subgrafo de G, e logo, pela Propriedade 2, tem largura em árvore no máximo k. □

Os Lemas 3.13 e 3.14 serão bastante utilizados na próxima seção, quando uma caracterização para grafos com largura em árvore no máximo 3 é apresentada.

k-árvores parciais

Grafos cordais, ou melhor, uma classe particular de grafos cordais, as k-árvores, podem ainda prover uma caracterização interessante para grafos de largura limitada. Considere inicialmente o seguinte resultado:

Teorema 3.15. *Uma k-árvore tem largura em árvore k.*

Prova. Uma k-árvore é cordal, e, portanto, admite uma decomposição em árvore mínima onde toda parte é uma clique. Todas as cliques da k-árvore tem tamanho $k + 1$, e logo a largura é também k. □

A caracterização segue do seguinte fato:

Lema 3.16. *Um grafo G tem largura em árvore no máximo k se e somente se G é uma k-árvore parcial.*

Prova. Se G é uma k-árvore, então a largura em árvore de G é k, pelo Teorema 3.15. Se G é uma k-árvore parcial então deve ser subgrafo de alguma k-árvore H, e logo, pela Propriedade 2, $LA(G) \leq LA(H) = k$. □

Através da caracterização sugerida no Lema 3.16, pode-se encontrar ainda uma propriedade que limita o número de arestas em um grafo com largura em árvore limitada:

Lema 3.17. *Se a largura em árvore de $G = (V, E)$ é no máximo k, então $|E| \leq k|V| - \frac{1}{2}k(k + 1)$.*

Prova. Seja $G = (V, E)$ uma k-árvore. Pelo Teorema 3.15, G admite uma decomposição em árvore $D = (\mathcal{X} = \{X_i\}_{i \in I}, T = (I, F))$ onde cada X_i contém uma clique de tamanho $k + 1$. É fácil ver, usando indução, que $|I| = |V| - k$, já que para quaisquer X_i e X_j tais que $X_i \cap X_j \neq \emptyset$, tem-se $|X_i \cap X_j| = k$. Se $|V| = k + 1$, então T tem somente um nó e $|E| = \frac{1}{2}k(k + 1)$. Se T tem $|I|$ nós, então foram acrescentados $|I| - 1$ vértices a K_{k+1}, feitos adjacentes a k vértices, de modo a obter G pela definição de k-árvore. Vale notar que, para cada vértice acrescentado, foram adicionadas k arestas.

Desta forma, todo vértice de G tem no mínimo k vizinhos — contribuindo com $k \cdot |V|/2$ arestas — e, considerando-se as novas arestas adicionadas ao longo da construção de G, tem-se uma contribuição de $k \cdot (|V| - k - 1)/2$ arestas. Logo,

$$|E| = \frac{k \cdot |V|}{2} + \frac{k \cdot (|V| - k - 1)}{2} = k \cdot |V| - \frac{1}{2}k(k+1).$$

Se G é um subgrafo de uma k-árvore, ou seja, uma k-árvore parcial, então claramente $E(G) \leq k \cdot |V| - \frac{1}{2}k(k+1)$. Além disso, pelo Lema 3.16, G tem largura em árvore no máximo k, e o lema segue. □

3.7 DEA para grafos com largura no máximo 3

Antes de iniciar uma discussão sobre uma caracterização por menores proibidos para a classe de grafos com largura em árvore no máximo 3, é importante considerar inicialmente o fato a seguir, de modo a perceber melhor, como nas caracterizações anteriores, a dificuldade em se obter uma DEA D de um grafo G tal que $LA(D) < LA(G)$.

Lema 3.18. *Os grafos M_6, M_8 e M_{10} da Figura 3.13 têm largura em árvore 4.*

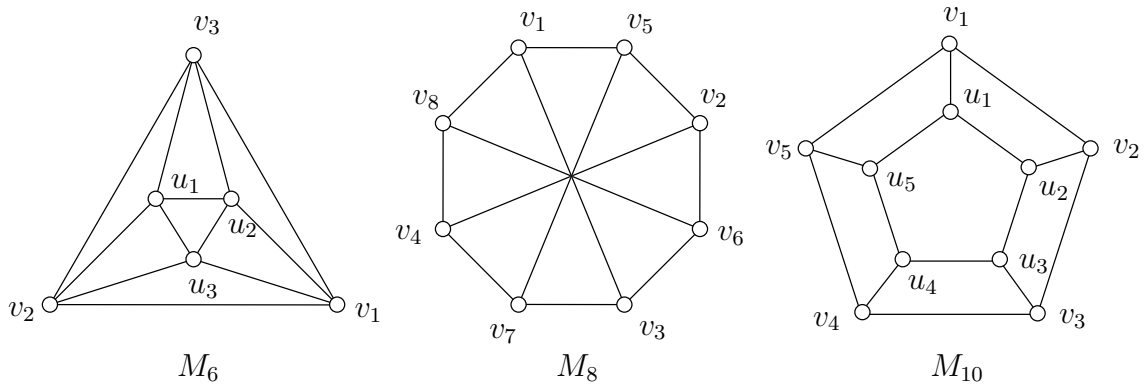


Figura 3.13. Os grafos M_6, M_8 e M_{10} têm largura em árvore 4.

Prova.

(M_6) Sejam u_1, u_2 e u_3 os vértices de M_6 pertencentes ao triângulo interno e v_1, v_2 e v_3 pertencentes ao triângulo externo, como na Figura 3.13.
Como v_1 e u_1 têm 4 vizinhos em comum, v_2, v_3, u_2 e u_3, então, pela Propriedade 8, ou M_6 tem largura em árvore no máximo 3 e (v_1, u_1) é necessária em M_6, ou $S_1 = \{v_2, v_3, u_2, u_3\}$ está contida numa parte de uma decomposição em árvore D de M_6. Neste último caso, pela Propriedade 7 (continência de subgrafo bipartite completo), tanto $\{v_1\} \cup S_1$ como $\{u_1\} \cup S_1$ pertencem a uma parte de D, e logo M_6 tem largura em árvore 4. Suponha que não, e logo (u_1, v_1) é necessária.
Pelo mesmo raciocínio, considerando agora v_2 e u_2, tem-se que (u_2, v_2) é necessária para largura em árvore 3 em M_6, ou tanto $\{u_2\} \cup S_2$ como $\{v_2 \cup S_2\}$, $S_2 =$

$\{v_1, v_3, u_1, u_3\}$, forçam a largura em árvore de M_6 para 4. Mas, pela necessidade de (u_1, v_1) e (u_2, v_2), tracejadas na Figura 3.14, tem-se que $S_3 = \{u_1, v_1, u_2, v_2\}$ deve pertencer a uma mesma parte de uma decomposição em árvore para M_6, e logo não há como impedir a largura em árvore de M_6 de ser 4, já que $\{v_3\} \cup S_3$ e $\{u_3\} \cup S_3$ formariam uma clique de tamanho 5 em qualquer 4-cordalização de M_6. A Figura 3.14 apresenta ainda uma decomposição em árvore de largura mínima 4 para M_6.

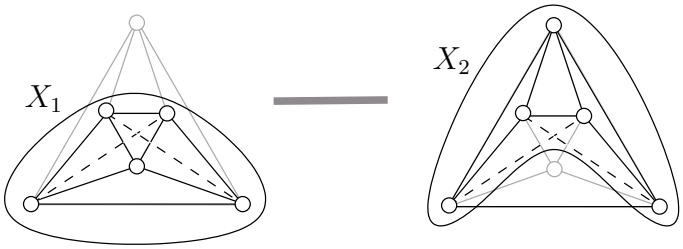


Figura 3.14. Decomposição em árvore do grafo M_6.

(M_8) Sejam $v_1, \ldots, v_8$ os vértices de M_8 como na Figura 3.13. Contraindo-se as quatro arestas (v_i, v_{i+4}), $1 \leq i \leq 4$, obtém-se um K_4, e logo M_8 tem largura em árvore no mínimo 3. Além disso, como cada vértice em M_8 tem três vizInhos, basta aplicar o resultado do Lema 3.14 fazendo S igual a $N_{M_8}(v)$ — o conjunto de vizinhos de qualquer vértice v — e logo, como $LA(M_8[\{v\} \cup S]) \leq 3$, não existem arestas necessárias para largura 3 em M_8 entre vértices não adjacentes.
No entanto, com a ajuda da simetria em M_8, podem-se estabelecer relações entre a presença de arestas em alguma 3-cordalização de M_8 e a ausência de outras. Obviamente essas relações devem ser pesquisadas entre vértices não adjacentes. Considere então $C = \{v_5, v_6, v_7, v_8\}$ composto pelos vértices em negrito na Figura 3.15, e os seguintes casos para construção de uma decomposição em árvore em M_8:

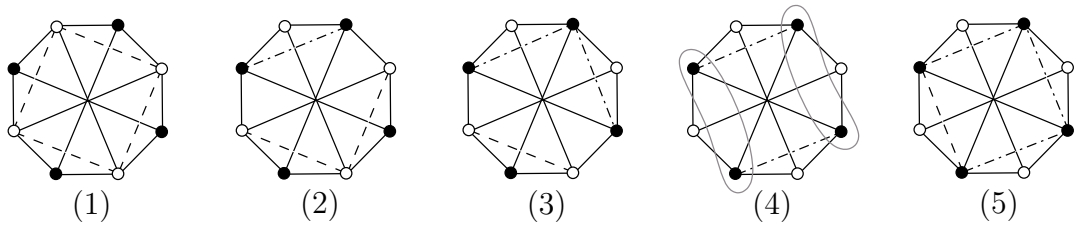


Figura 3.15. Casos para construção de uma decomposição em árvore de M_8 baseada em cordalização.

1. *Não existem arestas entre elementos de C.* Neste caso, para análise de necessariedade entre os pares de vértices não adjacentes de $C' = \{v_1, v_2, v_3, v_4\}$, deve-se encontrar, para cada par, um conjunto S satisfazendo o Lema 3.14 tal que S não contenha dois elementos de C — caso contrário, seria obtida uma decomposição em árvore cujas partes contém mais de dois elementos de C. Mas M_8 é 3-conexo, e logo $|S| \geq \kappa(M_8) = 3$ para que S seja um separador.

Além disso, $|S| \leq 4$ pelas condições do Lema 3.14 para $k = 3$, de modo que não é possível encontrar tal S para qualquer par de vértices em C', já que pelo menos dois vértices de S devem estar em C. Logo, qualquer par de vértices não adjacentes em C' deve ter uma aresta necessária, mostradas em tracejado na Figura 3.15(1). Não é difícil verificar que todas as decomposições em árvore de uma cordalização de M_8 satisfazendo a presença de arestas necessárias C e ausência de arestas em C' têm largura em árvore no mínimo 4. A Figura 3.16 ilustra uma decomposição em árvore de largura mínima de uma 4-cordalização de M_8, e logo também de M_8, respeitando C e C'.

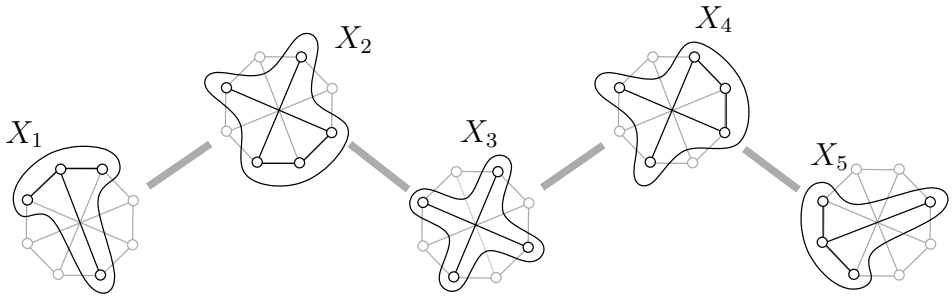


Figura 3.16. Decomposição em árvore do grafo M_8, caso 1.

2. *Existe apenas uma aresta necessária em C.* Seja (v_5, v_8) essa aresta, mostrada em estilo traço-ponto na Figura 3.15(2). Mesmo com (v_5, v_8) sendo necessária, não é possível encontrar um conjunto S com no máximo quatro vértices separando v_3 de v_2 e de v_4 e que não contenha dois vértices de C com exceção de v_5 e v_8. Logo, (v_2, v_3) e (v_3, v_4) são necessárias, sendo indicadas por arestas tracejadas na Figura 3.15(2). Novamente todas as cordalizações de M_8 obedecendo as condições deste caso com relação a C e C' são tais que o tamanho da maior clique é 5, e logo todas as decomposições em árvore mínimas têm largura 4. A Figura 3.17 ilustra uma destas decomposições; todas as outras resultam de simetria em M_8, ou seja, de variações na escolha da única aresta necessária em C.
3. *Existem duas arestas necessárias e adjacentes em C.* Sejam (v_5, v_8) e (v_5, v_6) tais arestas, ilustradas em traço-ponto na Figura 3.15(3). Considere agora o par (v_3, v_4). Qualquer v_3v_4-separador S deve conter v_7, já que este é vizinho comum do par. Mas S não pode conter v_5, v_6 e v_8, já que apenas as arestas entre eles são necessárias, e logo não existe S satisfazendo as condições do Lema 3.14 o que resulta na necessariedade de (v_3, v_4) (em tracejado na Figura 3.15(3)). Basta agora ver que qualquer cordalização obtida para esta caso será um *supergrafo* das cordalizações obtidas no caso 2 — aqui são permitidas arestas entre v_1 e v_2, v_2 e v_3, e v_1 e v_4, ao contrário do caso 2, onde não existem arestas entre v_5 e v_6, v_6 e v_7, e v_7 e v_8 — e logo toda decomposição em árvore para M_8 obtida aqui tem largura no mínimo 4. A decomposição ilustrada na Figura 3.17, por exemplo, satisfaz também as condições deste caso.

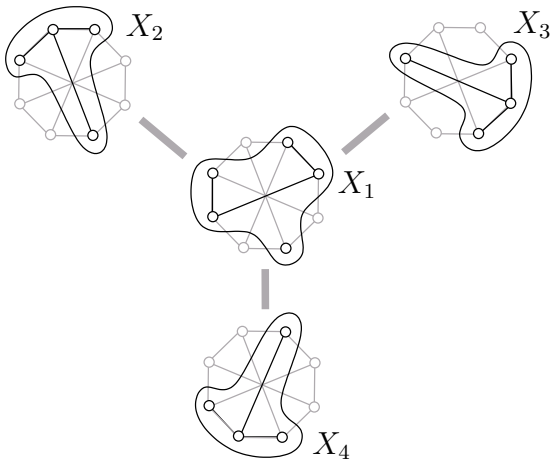


Figura 3.17. Decomposição em árvore do grafo M_8, caso 2.

4. *Existem duas arestas necessárias e não adjacentes em C.* Sejam (v_5, v_8) e (v_6, v_7) as arestas necessárias em C, traço-pontilhadas na Figura 3.15(4). Note agora que C induz um subgrafo bipartite completo nestas cordalizações de M_8, e logo, pela Propriedade 7, ou v_5 e v_6 devem pertencer a uma mesma parte em alguma decomposição em árvore de M_8 (satisfazendo as condições de arestas necessárias para este caso) ou v_7 e v_8 devem pertencer a uma mesma parte. Logo, tem-se uma contradição, porque apenas v_5 e v_8, e v_6 e v_7 podem ser necessárias pela hipótese do caso, e logo este não é viável.
5. *Existem três arestas necessárias em C.* Considere as arestas traço-pontilhadas na Figura 3.15(5), (v_5, v_8), (v_5, v_6) e $v_6, v_7)$, como sendo necessárias em C. Se (v_7, v_8) é necessária então qualquer cordalização de M_8 derivada é supergrafo de uma cordalização obtida no caso 1, tendo portanto largura em árvore no mínimo 4. Neste caso, a decomposição em árvore da Figura 3.16 é valida também para este caso.
   Seja então M'_8 uma cordalização de M_8 em que todos os vértices em C são adjacentes entre si, com exceção de v_7 e v_8. Note que se existe uma aresta entre v_4 e v_5, então as contrações de (v_1, v_8), (v_2, v_6) e (v_3, v_7) formam um K_5, e logo suponha ainda que v_4 e v_5 não são adjacentes em M'_8, assim como, por simetria, v_4 e v_6, v_2 e v_7, e v_2 e v_8. Mas isso não é possível, pois desta forma v_2, v_4, v_8 e v_5 formam um ciclo induzido, por exemplo, e M'_8 não é uma cordalização. Portanto, (v_7, v_8) é necessária.

Pela simetria de M_8, os casos acima resultam em todas as combinações possíveis para cordalizações mínimas de M_8. Logo, como todas são 4-cordalizações, segue pelo Corolário 3.11 que $LA(M_8) = 4$.

(M_{10}) Sejam v_i, u_i, $1 \leq i \leq 5$, os vértices de M_{10}, como mostrado na Figura 3.13. A contração de todos os u_i e de uma aresta qualquer do ciclo externo resulta num K_4, e logo $LA(M_{10}) \geq 3$. Assim como no M_8, todo vértice possui 3 vizinhos, e logo, não é possível encontrar, para qualquer par de vértices não adjacentes em M_{10}, um conjunto S satisfazendo as condições do Lema 3.14; desta forma, não existem

arestas necessárias entre vértices não adjacentes. A estratégia então é semelhante a utilizada para o M_8, mas como a enumeração dos casos seria mais longa, ela será aqui suprimida, e será apenas assumido que $LA(M_{10}) \geq 4$.
Basta portanto encontrar uma decomposição em árvore de largura mínima 4 para concluir a prova. Considere então o par de vértices (v_1, v_3), e 3 caminhos disjuntos $P_1 = \langle v_1, v_5, v_4, v_3 \rangle$, $P_2 = \langle v_1, u_1, u_2, u_3, v_3 \rangle$ e $P_3 = \langle v_1, v_2, v_3 \rangle$ em M_{10}. Sejam $U_1 = \{v_4, v_5\}$, $U_2 = \{u_1, u_2, u_3\}$ e $U_3 = \{v_2\}$ conjuntos contendo os vértices internos de cada caminho, respectivamente. Então, pelo Lema 3.13 e pelo fato de que $LA(M_{10}) > 2$, existe um menor M'_{10} de M_{10} obtido pela contração de cada U_i em novos vértices v_{U_i}, $1 \leq i \leq 3$, tal que os v_{U_i} pertencem a uma mesma parte de uma decomposição em árvore de M'_{10}. Na Figura 3.18(a), podem-se ver os caminhos P_i, os conjuntos U_i e M'_{10}.

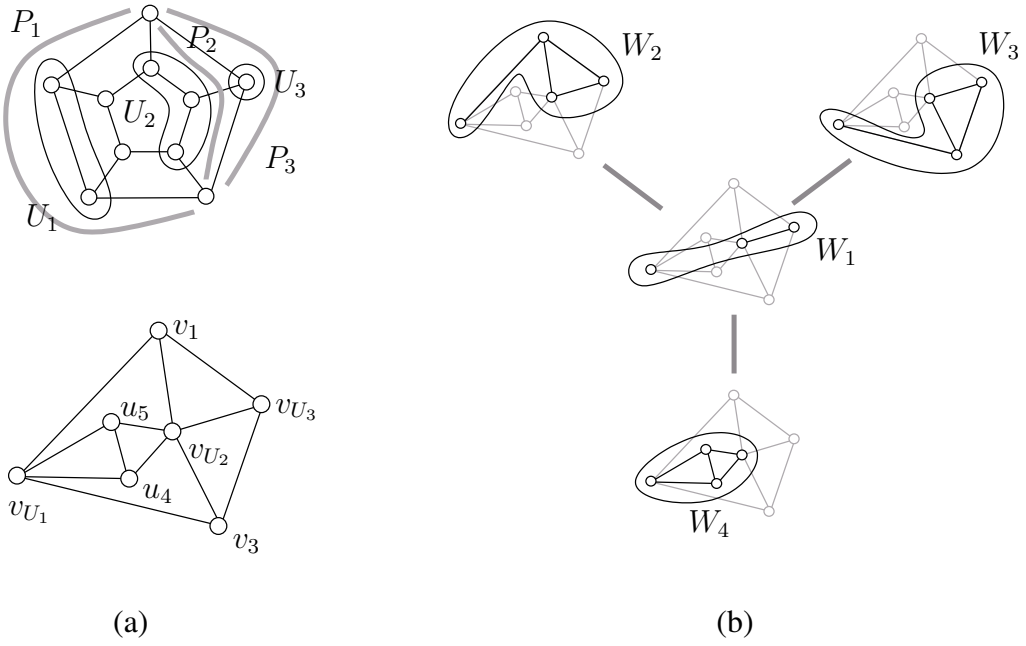


Figura 3.18. Decomposição em árvore de um menor do grafo M_{10}.

Como a contração de $\{u_4, u_5, v_{U_2}\}$ resulta num K_4, então $LA(M'_{10}) > 2$. Na Figura 3.18(b), pode-se ver uma decomposição em árvore mínima $D = (\mathcal{W} = \{W_i\}_{1 \leq i \leq 4}, T = (I, F))$ de M'_{10} com largura 3, na qual $\{v_{U_1}, v_{U_2}, v_{U_3}\}$ estão numa mesma parte. Pela aplicação do Lema 3.2, pode-se derivar uma decomposição em árvore $D' = (\mathcal{W}' = \{W'_i\}_{i \in I}, T = (I, F))$ para M_{10} onde cada parte W'_i corresponde a união dos $f(v)$ para $v \in W_i$, como definido no Lema. Claramente, a largura de D' é 6. A partir de D', pode-se ainda derivar uma outra decomposição em árvore D^* para M_{10}, eliminando o excesso de cada W'_i e procurando manter a estrutura em árvore. As partes W'_1, W'_2 e W'_3 podem ser substituídas por X_1, X_2, X_3 e X_4, com menos vértices cada e cobrindo os mesmos vértices e arestas, como ilustra a Figura 3.19. Além disso, W'_4 pode ser substituído por X_5 e X_6, de onde se obtém uma decomposição em árvore de largura 4.

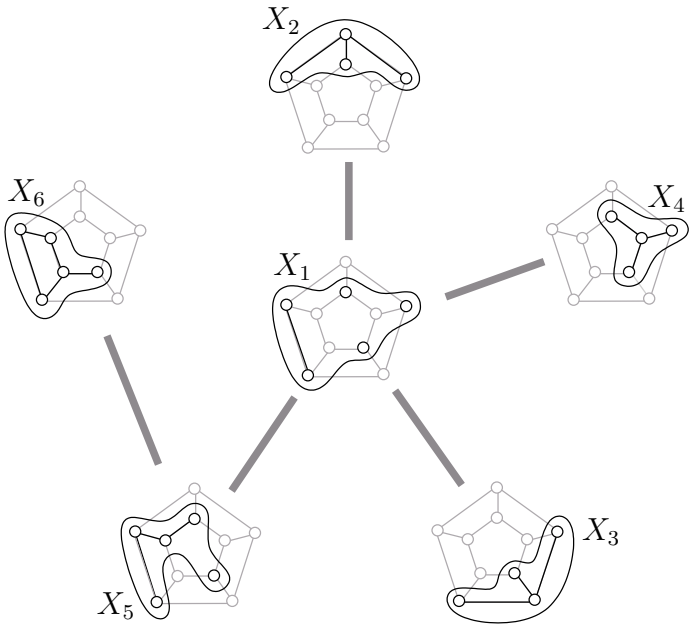


Figura 3.19. Decomposição em árvore do grafo M_{10}.

□

A seguinte caracterização pode ser obtida, sendo aqui exposta sem prova:

Teorema 3.19 (Arnborg et al. [1993]). *A classe dos grafos com largura em árvore no máximo 3 é exatamente a classe de grafos que tem K_5, M_6, M_8 e M_{10} como menores proibidos, Proib$_{\preccurlyeq}(\{K_5, M_6, M_8, M_{10}\})$.*

É importante notar que, ao contrário das classes de grafos com largura em árvore no máximo k, $k = 1, 2$, não basta proibir K_{k+2} como menor; ou seja, não ter um K_5 como menor é uma condição necessária para que a largura de um grafo seja no máximo 4, mas não suficiente. Os motivos dessa extensão necessária no conjunto de obstrução — que está longe de ser tão óbvia como nas classes de grafos com largura limitada vistas até agora — apenas serão melhor compreendidos ao se analisar mais profundamente a estrutura dos grafos com largura em árvore limitada, e não tão pequena. Este é o objetivo do resto do capítulo.

3.8 Conjuntos k-ligados, arbustos e novelos

Na Seção 2.4, foi apresentada a característica de um grafo ser k-ligado, e viu-se que esta definição é mais forte que a de k-conectividade. Na verdade, a k-conectividade em um grafo representa um parâmetro local do quão conexo um grafo é, baseado nas regiões menos conexas do grafo. Desta forma, pode-se aproveitar o conceito de k-ligado, redefinindo-o de modo a refletir uma medida mais global de conectividade.

Definição 15 (Conjunto k-ligado e componente grande). Um subconjunto de vértices S de um grafo $G = (V, E)$ é dito *k-ligado* se, para qualquer conjunto X, $|X| < k$,

existe uma componente (única) de $G - X$ contendo mais da metade dos vértices de S, chamada de *componente grande* (de $G - X$).

Vale notar que para um conjunto S ser k-ligado, deve-se ter necessariamente $|S| \geq 2k$ — caso contrário, basta tomar um subconjunto de S com mais de k vértices para que não exista uma componente grande de S. Por outro lado, é fácil ver que qualquer conjunto S pode ser no máximo $|S|/2$-ligado.

Definição 16 (Ligação). A *ligação* de G, denotada por $\text{lig}(G)$, é o maior inteiro para o qual G contém um conjunto k-ligado.

Para perceber que a ligação de um grafo é uma medida mais global de conectividade, considere uma grade $r \times s$. Uma *grade* de ordem $r \times s$ é um grafo tendo como conjunto de vértices $\{1, \ldots, r\} \times \{1, \ldots, s\}$ e conjunto de arestas definido por:

$$\{((i, j), (i', j')) \, : \, |i - i'| + |j - j'| = 1\}.$$

As *cruzes* da grade são os $r \cdot s$ conjuntos definidos por:

$$C_{i,j} = \{(i, l) \, : \, l = 1, \ldots, s\} \cup \{(l, j) \, : \, l = 1, \ldots, r\}$$

ou seja, cada cruz $C_{i,j}$ corresponde a união, na grade, da i-ésima linha e da j-ésima coluna. Uma grade e duas de suas cruzes são exemplificadas na Figura 3.20. Pode-se ver facilmente que toda grade é planar.

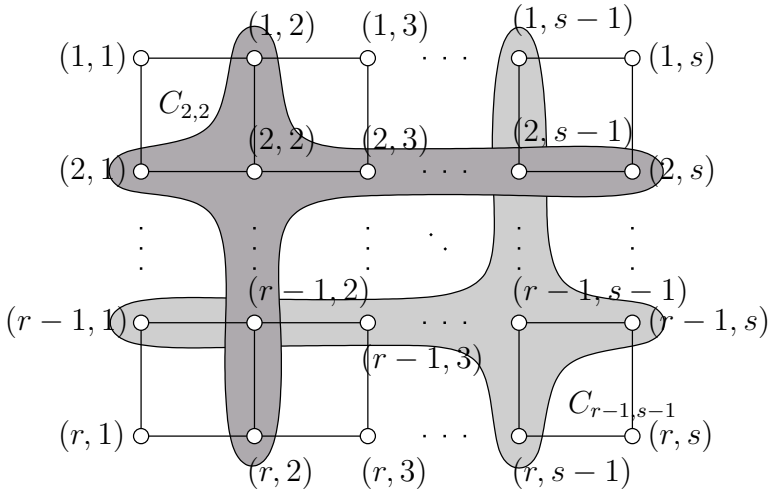


Figura 3.20. Uma grade $r \times s$ e as cruzes $C_{2,2}$ e $C_{r-1,s-1}$.

Embora toda grade G de ordem $r \times s$ seja 2-conexa e de grau máximo 4, pode-se ver que qualquer cruz C de G é um conjunto k-ligado, onde $k = \min\{r, s\}$. Para tanto, basta perceber que apenas a retirada de no mínimo k vértices pode separar C — correspondentes a uma linha, se $r \leq s$, ou a uma coluna, caso contrário. Suponha que $r \leq s$, e logo, no pior caso, a retirada de $r - 1$ vértices da linha completamente coberta por C faz com que uma componente de G tenha pelo menos $s - 1$ vértices, ou seja, mais do

que $(r + s - 1)/2 = |C|/2$. Desta forma, a ligação de uma grade pode ser arbitrariamente grande dependendo das dimensões desta, em contraste ao conceito convencional de conectividade.

Um outro conceito relacionado à uma nova medida de conectividade é o de *arbusto*.

Definição 17 (Arbusto). Seja $G = (V, E)$ um grafo. Dois subconjuntos X, Y de vértices de V *tocam-se* se eles se interceptam ($X \cap Y \neq \emptyset$) ou se existe uma aresta $e = (x, y)$ em G entre eles ($x \in X$ e $y \in Y$). Uma família de subconjuntos de G que se tocam dois a dois é chamada um *arbusto*.

Os dois conceitos estão bastante relacionados. Se S é um conjunto k-ligado, então pode-se ver que, para cada par de conjuntos X e Y, $|X|, |Y| < k$, as componentes grandes de $G - X$ e $G - Y$ se interceptam. Logo, a família $\mathcal{B}$ de todas as componentes grandes com relação a S em G é um arbusto. Como cada conjunto $X \subseteq V(G)$, $|X| < k$, gera uma componente grande em $\mathcal{B}$, não há como um conjunto com menos de k vértices interceptar todos os elementos em $\mathcal{B}$. Assim como um arbusto pode ser obtido a partir de um conjunto k-ligado, a recíproca também é verdadeira. Tal aspecto pode ser melhor explorado com a introdução de mais um conceito:

Definição 18 (Conjunto de acerto). Um conjunto de vértices U *cobre* um arbusto $\mathcal{B}$ se os vértices em U encontram cada elemento de $\mathcal{B}$; neste caso, U é dito um *conjunto de acerto* para $\mathcal{B}$.

Através da definição de conjunto de acerto, pode-se ainda conferir um caráter quantitativo a um arbusto:

Definição 19 (Número de arbusto). A *ordem de um arbusto* $\mathcal{B}$ equivale a cardinalidade do seu menor conjunto de acerto, sendo denotado por $\|\mathcal{B}\|$. O *número de arbusto* de um grafo G, denotado por $NA(G)$, é o máximo obtido dentre as ordens de seus arbustos.

Logo, pela definição de número de arbusto, pode-se dar uma definição mais exata para a relação entre arbusto e conjunto k-ligado. Como todo conjunto k-ligado gera um arbusto $\mathcal{B}$ tal que este não pode ser coberto por nenhum conjunto X com menos de k vértices, segue que $\mathcal{B}$ tem ordem no mínimo k.

Por outro lado, qualquer conjunto de acerto B^* mínimo de um arbusto $\mathcal{B}$ pode gerar um conjunto no máximo $\|\mathcal{B}\|/2$-ligado. Basta ver que um conjunto $X \subseteq B^*$, $X \leq \|\mathcal{B}\|/2$, gera uma componente de $\mathcal{B}$ que deve conter os vértices restantes de B^*, sendo portanto uma componente grande. Logo, todo arbusto gera um conjunto no mínimo k-ligado tal que k é no máximo metade da ordem do arbusto. O lema seguinte vincula as relações para todo o grafo, e segue como conseqüência direta do que foi discutido nesta seção até agora.

Lema 3.20 (Reed [1997]). *Para todo grafo G, lig*$(G) \leq NA(G) \leq 2$*lig*(G).

Prova. Como conjunto S k-ligado gera um arbusto $\mathcal{B}$ de ordem no mínimo k, o maior valor de possível para a ordem de um arbusto em um grafo G depende do maior k para o qual G admite um conjunto k-ligado, ou seja, $NA(G) = \max\{\|\mathcal{B}\| : \mathcal{B}$ é um arbusto em $G\} \geq \max\{k :$ existe S k-ligado em $G\}$ =lig(G).

O outro lado da desigualdade é mostrado de modo semelhante. Seja $\mathcal{B}$ um arbusto que realiza o número de arbusto em G, ou seja, $\|\mathcal{B}\| = NA(G)$. Como todo arbusto gera

um conjunto k-ligado, $\mathcal{B}$ garante a existência de um tal conjunto com k no mínimo $\|\mathcal{B}\|/2$. Logo, o maior valor de k depende de $NA(G)$, $\text{lig}(G) = \max\{k : \text{existe } S \ k\text{-ligado em } G\} \geq NA(G)/2$. □

Um exemplo típico de arbusto é o conjunto de *cruzes* em uma *grade*. Claramente, a união das cruzes de uma grade forma um arbusto de ordem $\min\{r, s\}$, onde qualquer linha e qualquer coluna representa um conjunto de acerto. De forma a aproveitar o fato de que a menor dimensão de uma grade é a mais relevante, normalmente refere-se a uma grade regular $k \times k$. Claramente, uma grade $k \times k$ tem um arbusto de ordem k.

De maneira semelhante aos conceitos de arbusto e conjunto k-ligado, assim também um arbusto em um grafo G e uma decomposição em árvore para G estão intimamente relacionados. O modo como essa relação se dá é elucidado pelos dois lemas seguintes.

Lema 3.21 (Reed [1997]). *Sejam G um grafo, $\mathcal{B}$ um arbusto de G e $D = (\mathcal{X} = \{X_i\}_{i \in I}, T = (I, F))$ uma decomposição em árvore de G. Então existe um nó b em T tal que X_b é um conjunto de acerto para $\mathcal{B}$.*

Prova. Sejam $G = (V, E)$, D e $\mathcal{B} = \{B_j\}_{j \in J}$ como definidos no enunciado. Para cada B_j em $\mathcal{B}$, seja $T_{B_j} = \{t \in T : v \in B_j \text{ e } v \in X_t, v \in V\}$. Claramente, cada T_{B_j} induz uma sub-árvore de T, e logo, pelo Lema 2.2, $T^* = \bigcap_{j \in J} T_{B_j} \neq \emptyset$ induz uma nova sub-árvore de T. Basta agora ver que qualquer nó b de T^* pertence a todas as sub-árvores T_{B_j}, e logo X_b encontra todos os elementos de $\mathcal{B}$, sendo um conjunto de acerto para este. □

Lema 3.22 (Diestel [2000]). *Para todo arbusto $\mathcal{B}$ de um grafo G existe uma decomposição em árvore $D = (\mathcal{X} = \{X_i\}_{i \in I}, T = (I, F))$ de G tal que se $|X_i| > \|\mathcal{B}\|$, $X_i \in \mathcal{X}$, então X_i não cobre $\mathcal{B}$.*

Uma decomposição em árvore satisfazendo o Lema 3.22 acima é dita $\mathcal{B}$-*admissível*. A prova do lema acima é na verdade uma parte da prova de outro teorema central na relação entre arbustos e DEA's; a versão da prova fornecida por Diestel é mais curta que a prova original.

Teorema 3.23 (Seymour e Thomas [1993]). *Seja $k \geq 0$ um inteiro. Um grafo tem largura em árvore no máximo k se e somente se contém um arbusto de ordem (estritamente) maior que k.*

O Teorema 3.23 é conhecido como *Teorema da Dualidade para Largura em Árvore* sendo também usualmente expresso sob outra forma:

Teorema 3.24 (Dualidade para Largura em Árvore, Robertson e Seymour [1986]). *Para todo grafo G, $LA(G) = NA(G) - 1$.*

Prova. Seja G um grafo. Pelo Lema 3.21, a partir de uma decomposição em árvore D de G é sempre possível obter um arbusto $\mathcal{B}$ tal que existe uma parte de D contendo um conjunto de acerto mínimo para $\mathcal{B}$. Logo, $LA(D) \geq \|\mathcal{B}\| - 1$. Como a largura em árvore de qualquer decomposição é limitada pela ordem do seu arbusto derivado, segue que o menor valor para a largura é no mínimo o maior valor para ordem, e logo $LA(G) \geq NA(G) - 1$.

Por outro lado, pelo Lema 3.22, todo arbusto $\mathcal{B}$ gera uma decomposição em árvore $\mathcal{B}$-admissível D, e logo, como qualquer parte de D encontra todo elemento de

$\mathcal{B}$, $LA(D) \leq \|\mathcal{B}\| - 1$. Pelo mesmo motivo, a largura em árvore de G deve estar limitada pelo maior valor possível de um conjunto de acerto para um arbusto $\mathcal{B}$, e logo segue que $LA(G) \leq NA(G) - 1$. □

Embora a relação entre arbustos e DEA's seja bastante estreita, ela não permite uma equivalência entre os conceitos: um arbusto $\mathcal{B}$ pode gerar mais de uma decomposição em árvore de largura no máximo $\|\mathcal{B}\|-1$, enquanto que uma decomposição em árvore D pode gerar mais de um arbusto de ordem no máximo $LA(D) + 1$. Para se obter uma relação mais estreita ainda, é preciso acrescentar mais condições à formação de um arbusto.

Definição 20 (Novelo). Um arbusto $\mathcal{N}$ é um *novelo* se, para qualquer tripla (N_1, N_2, N_3) de elementos de $\mathcal{N}$, uma das duas situações abaixo ocorre:

1. $N_1 \cap N_2 \cap N_3$ é não vazio;
2. existe uma aresta e tal que T_1, T_2 e T_3 contém uma extremidade de e.

Dados um arbusto $\mathcal{B}$ e um grafo G, define-se a função $f_{\mathcal{B}} : \{X : X \subseteq V(G), |X| < \|\mathcal{B}\|\} \mapsto \mathcal{P}(V(G))$ como um mapeamento de um subconjunto X satisfazendo o domínio de $f_{\mathcal{B}}$ à única componente de $G - X$ contendo um elemento de $\mathcal{B}$. Diz-se então que dois arbustos $\mathcal{B}_1$ e $\mathcal{B}_2$ são *distinguíveis* em um grafo G se existe algum $X \subseteq V(G)$, $|X| < \|\mathcal{B}_1\|$, $|X| < \|\mathcal{B}_2\|$, tal que $f_{\mathcal{B}_1}(X) \neq f_{\mathcal{B}_2}(X)$ — e, neste caso, X é chamado $(\mathcal{B}_1, \mathcal{B}_2)$-*distintor*; caso contrário, $\mathcal{B}_1$ e $\mathcal{B}_2$ são ditos *indistinguíveis*.

Um arbusto é dito *maximal* se não existe nenhum outro indistinguível dele e que tenha maior ordem. O mesmo conceito vale para novelos; de fato, vale notar que novelos maximais indistinguíveis têm mesma ordem.

O teorema abaixo estabelece uma relação muito próxima entre os novelos maximais de um grafo G e uma decomposição em árvore especial de G.

Teorema 3.25 (Decomposição em árvore canônica, Robertson e Seymour [1995]). *Para qualquer grafo G, pode-se construir uma decomposição em árvore $D = (\mathcal{X}, T)$ que tenha as seguintes propriedades:*

1. *Para cada novelo maximal $\mathcal{N}$ de G, existe exatamente um nó $t(\mathcal{N})$ de T tal que $X_{t(\mathcal{N})}$ forma um conjunto de acerto para $\mathcal{N}$.*
2. *Se $\mathcal{N}_1$ e $\mathcal{N}_2$ são novelos maximais indistinguíveis então $t(\mathcal{N}_1) = t(\mathcal{N}_2)$.*
3. *Se $\mathcal{N}_1$ e $\mathcal{N}_2$ são novelos maximais distinguíveis então $t(\mathcal{N}_1) \neq t(\mathcal{N}_2)$ e existe uma aresta st no caminho $t(\mathcal{N}_1)Tt(\mathcal{N}_2)$ tal que $X_s \cap X_t$ é um $(\mathcal{N}_1, \mathcal{N}_2)$-distintor de ordem mínima.*
4. *Para cada nó t de uma decomposição em árvore, existe um novelo maximal tal que $t(\mathcal{N}) = t$.*

Como Reed (1997) observa, o Teorema 3.25 garante a existência de uma decomposição em árvore D para um grafo G que separa G em partes altamente conexas, usando separadores que são os menores possíveis. Para isto, basta ver que existe uma relação de equivalência entre D e um conjunto de novelos maximais distinguíveis N para G onde, pelas restrições (1) e (4) do teorema, cada parte de D é um conjunto de acerto

para algum novelo em N. Logo, não podem existir partes X_i e X_j em D tais que $X_i \subseteq X_j$, e conseqüentemente D é uma boa decomposição em árvore.

3.9 Grades

Como mostrado na seção anterior, as grades tornam-se interessantes por apresentarem ligação e número de arbusto arbitrariamente grandes. Também na seção anterior foi apresentada a relação próxima entre arbustos e decomposições em árvore, melhor explicitada pelos Teoremas 3.23 e 3.23. Em particular, pode-se ver que uma grade $k \times k$ tem largura em árvore no mínimo $k-1$, já que apresenta um arbusto de ordem k. De fato, a largura em árvore de uma grade $k \times k$ é exatamente k (em (Reed 1997) é mostrada uma decomposição em árvore de largura mínima).

Desta forma, uma grade $k \times k$ pode ter largura em árvore arbitrariamente grande, e logo torna-se uma obstrução natural para classes de grafos com largura em árvore limitada: se um grafo G possui uma grade $k \times k$ como menor, então, pela Propriedade 4, a largura em árvore de G é no mínimo k. Uma conseqüência deste fato é que, ao se construir um conjunto de obstrução para uma classe de grafos com largura em árvore limitada, as grades e seus menores têm importância considerada. De fato, como toda grade e seus menores são planares, qualquer classe $\text{Proib}_{\preccurlyeq}(\mathcal{H})$, onde algum elemento H de $\mathcal{H}$ é não planar, contém todas as grades, e logo não pode conter grafos com largura em árvore limitada. O seguinte teorema estabelece uma equivalência entre o que foi discutido:

Teorema 3.26 (Robertson e Seymour [1986]). *Dado um grafo H, os grafos que não contém H como menor têm largura em árvore limitada se e somente se H é planar.*

Uma prova do Teorema 3.26 bem mais curta que a original pode ser encontrada em (Diestel 2000). A idéia da prova apresentada por Diestel é mostrar que a proibição de qualquer grafo planar como menor limita a largura em árvore de um grafo. Além disso, basta mostrar tal proibição para uma grade, já que todo grafo planar é menor de alguma grade. Logo, basta mostrar que:

Teorema 3.27 (Robertson e Seymour [1986]). *Para todo inteiro r, existe um inteiro k tal que todo grafo com largura em árvore no mínimo k tem um menor de uma grade $r \times r$.*

Com base nestes resultados, torna-se mais claro agora o motivo da extensão do conjunto de menores proibidos para a classe de grafos G tais que $LA(G) \leq 3$. Não se pode proibir apenas o K_5, já que este é planar e logo $\text{Proib}_{\preccurlyeq}(\{K_5\})$ contém todas as grades, tendo seus grafos largura em árvore ilimitada.

3.10 Conceitos relacionados

Uma *decomposição em caminho* de um grafo G é uma decomposição em árvore $D = (\mathcal{X} = \{X_i\}_{i \in I}, T = (I, F))$ onde T é um *caminho*. Analogamente à definição de largura em árvore, define-se a *largura em caminho* de uma decomposição em caminho D como a

maior cardinalidade de suas partes menos um, $LC_G(D) = \max_{i \in I}\{|X_i| - 1\}$, e a largura em caminho de um grafo G como sendo a largura mínima observada dentre todas as larguras em caminho de G.

Decomposições em caminho são, em geral, de tratamento mais fácil, dada a simplicidade de sua estrutura. Como é de se esperar, várias propriedades que valem para decomposições em árvore valem também para decomposições em caminho, como as Propriedades 3, 2 e 4. Como se verá no próximo capítulo, decomposições em caminho podem inclusive ser usadas como base para o desenvolvimento de algoritmos para decomposição em árvore, assim como a pesquisa para tais algoritmos é enriquecida com o desenvolvimento de técnicas de decomposições em árvore mínimas para classes específicas de grafos.

Um outro conceito interessante é o de decomposição em árvore *ligada* ou *enxuta*:

Definição 21 (Decomposição em árvore ligada). Uma decomposição em árvore $D = (\mathcal{X} = \{X_i\}_{i \in I}, T = (I, F))$ de um grafo $G = (V, E)$ é dita *ligada* se, além das restrições usuais de uma decomposição em árvore, esta satisfaz:

[P4L] dados um número $s \in \mathbb{N}$ e nós $i, j \in I$ de T, ou G contém s caminhos disjuntos V_i-V_j ou existe um $t \in iTj$ tal que $|V_t| < s$.

A propriedade de ser ligada pode ser vista como um fortalecimento do conceito de ser boa: além de não conter partes redundantes, os ramos de uma decomposição em árvore ligada de um grafo G contém vértices de modo somente a manter a conectividade mínima em G. Desta forma, G será sempre minimalmente conexo ao longo de cada ramo. Decomposições em árvore ligadas também constituem ferramentas importantes para encontrar a largura em árvore de um grafo G, devido principalmente ao seguinte resultado:

Teorema 3.28 (Thomas [1990]). *Todo grafo G admite uma decomposição em árvore ligada de largura $LA(G)$.*

Intuitivamente, pode-se perceber que a busca por partes pequenas em uma decomposição pode vantajosamente ser compatibilizada com a identificação de subgrafos altamente conexos em um grafo, e que comporiam um ramo numa decomposição em árvore ligada.

Capítulo 4

Algoritmos de Decomposição em Árvore

4.1 Introdução

No capítulo anterior foram vistos conceitos e resultados teóricos envolvendo decomposições em árvore de grafos; o objetivo deste capítulo é complementar, e concentra-se em discutir técnicas e algoritmos para obtenção de decomposições em árvore. Em particular, um dos grandes objetivos na pesquisa de decomposições em árvore envolve o desenvolvimento de métodos para determinação da decomposição em árvore mínima de um grafo. Tal objetivo pode ser alcançado com a resolução do seguinte problema de otimização:

Problema: MIN LARGURAEMÁRVORE
Instância: Um grafo $G = (V, E)$.
Encontrar: A largura em árvore de G, $LA(G)$.

Um outro problema, mais simples, decide, dados um grafo G e um inteiro k, se a largura em árvore de G é no máximo k. Este corresponde a versão de decisão do problema acima, sendo denotado por k-LARGURAEMÁRVORE:

Problema: k-LARGURAEMÁRVORE
Instância: Um grafo $G = (V, E)$ e um inteiro k.
Decidir: A largura em árvore de G, $LA(G)$, é no máximo k?

Conforme Arnborg et al. (1987) mostraram, k-LARGURAEMÁRVORE é NP-completo, mesmo quando restrito a algumas classes de grafos, como grafos bipartite, por exemplo. No entanto, existem ainda outras classes de grafos para as quais k-LARGURAEMÁRVORE admite complexidade em tempo polinomial, como para as classes de grafos cordais. O objetivo de tratar o problema restrito a algumas classes de grafos é a obtenção de heurísticas para limitantes superiores, como se verá na Seção 4.2, onde grafos cordais são o foco.

Quando o parâmetro k é uma constante fixa, e logo não faz parte da entrada do problema, tem-se um caso bem estudado. Neste caso, distinguem-se duas versões do problema: a versão de *decisão*, quando se deve apenas decidir se a largura em árvore de um grafo G é no máximo k, retornando verdadeiro em caso afirmativo e falso caso contrário; e a versão de *construção*, quando também uma decomposição em árvore de largura no máximo k deve compor a saída, em caso afirmativo.

O primeiro algoritmo em tempo polinomial para a versão construtiva de k-LARGURAEMÁRVORE se deve a Arnborg et al. (1987), tendo complexidade $O(n^{k+2})$. A partir dos resultados com menores de grafos, Robertson e Seymour fornecem uma prova não construtiva da existência de um algoritmo em tempo $O(n^2)$ para a versão de decisão (Robertson e Seymour 1995). Seu algoritmo tem uma estrutura em duas fases:

1. Na primeira fase, um algoritmo em tempo $O(n^2)$ decide se G admite uma decomposição em árvore D com largura no máximo $4k + 3$ ou, caso contrário, se G tem largura em árvore maior que k. Como Bodlaender (1997) observa, na verdade Robertson e Seymour usam um conceito semelhante a largura em árvore para alcançar este resultado, mas a diferença é somente técnica e não relevante.

2. A decomposição obtida na primeira fase é agora usada para verificar, em tempo linear, se G contem algum menor pertencente ao conjunto de obstrução para a classe de grafos com largura em árvore no máximo k.

Grande parte dos algoritmos empregados para o problema ainda seguem, de alguma forma, a estrutura em duas fases. Como a complexidade de um algoritmo deste tipo depende da primeira fase, a grande maioria dos esforços de melhoria foram orientados para esta etapa. Vários autores forneceram versões mais eficientes, incluindo um algoritmo aleatório de Matousek e Thomas (1992), uma versão paralela de Lagergren (1996), e uma versão utilizando separadores com características especiais de Reed (1992), de complexidade $O(n \log n)$. Este último algoritmo será visto em maiores detalhes na Seção 4.3.

Cada um destes algoritmos de primeira fase determina se a largura em árvore de um grafo G dado como entrada é maior que k ou, caso contrário, retorna uma decomposição em árvore de G com largura no máximo $f(k)$, onde f é uma função linear de k com coeficientes pequenos. Para a segunda fase, Bodlaender e Kloks (1996) encontraram um algoritmo em tempo linear para a versão de construção, ou seja, dada uma decomposição em árvore de um grafo de entrada G com largura $l > k$, o algoritmo decide se G admite uma nova decomposição em árvore de largura no máximo k, e a retorna. Usando este resultado, Bodlaender (1996) encontrou uma algoritmo resolvendo as duas fases conjuntamente em tempo linear, para cada k constante, para a versão construtiva. Este é o melhor resultado encontrado até agora, sendo discutido na Seção 4.4.

Uma outra abordagem interessante foi desenvolvida por Arnborg et al. (1993), usando uma técnica conhecida como *redução em grafos*. Embora os algoritmos apresentados por Arnborg et al. tenham complexidade em tempo linear, sua complexidade em espaço é exponencial. Vários outros autores adotaram abordagens semelhantes como paradigma, como por exemplo Courcelle (1990), contribuindo para o refinamento da técnica. de Fluiter e Bodlaender (1996) modificaram alguns dos conceitos para obter um algo-

ritmo baseado em redução e com complexidade linear tanto em tempo como em espaço. A técnica de redução em grafos é apresentada na Seção 4.3.

Todos os algoritmos até agora apresentados têm uma constante escondida na expressão de complexidade que é, no mínimo, exponencial em k. Como Bodlaender (1997) comenta, a medida em que k aumenta os algoritmos tornam-se cada vez mais inadequados para aplicações práticas. Nos casos em que $k = 2$, 3 ou 4, algoritmos específicos foram encontrados, sendo de complexidade linear e baseados em redução em grafos (Matousek e Thomas 1991; Arnborg e Proskurowski 1986; Sanders 1996).

4.2 DEA em grafos cordais

Assim como no capítulo anterior uma ênfase especial foi dada aos grafos cordais, de onde algumas caracterizações importantes foram obtidas, é interessante discutir aqui, inicialmente, o problema de largura em árvore quando restrito a esta classe de grafos.

Antes de apresentar um algoritmo para discussão, é necessário introduzir a notação seguinte. Se $D = (\mathcal{X}, T)$ é uma decomposição em árvore de um grafo G, então $T(D)$ retorna a árvore da decomposição, ou seja, $T(D) = T$; de modo análogo, $\mathcal{X}(D)$ retorna a família de partes de D, $\mathcal{X}(D) = \mathcal{X}$. A função COMPONENTES$(G)$ retorna uma lista contendo os conjuntos de vértices que induzem cada componente no grafo G, respectivamente (um conjunto em cada posição da lista). O algoritmo DEA_CORDAL da Figura 4.1 constrói uma decomposição em árvore mínima para um grafo cordal.

```
▷ Algoritmo DEA_CORDAL(G)
▷ Entrada: Um grafo cordal G = (V, E).
▷ Saída: Uma decomposição em árvore D = (𝒳, T) mínima de G.

1.  início
2.      se G é completo então
3.          retorna ({V}, ({t}, ∅))
4.      senão
5.          selecione u, v : (u, v) ∉ E, u, v ∈ V
6.          S ← SEP_MINIMAL(u, v, G)
7.          D ← ({S}, ({t_S}, ∅))
8.          para cada C ∈ COMPONENTES(G[V \ S]) faça
9.              D' ← DEA_CORDAL(G[C ∪ S])
10.             selecione t' : S ⊆ X_t', t' ∈ T(D'), X_t' ∈ 𝒳(D')
11.             𝒳(D) ← 𝒳(D) ∪ 𝒳(D')
12.             T(D) ← T(D) ∪ T(D')
13.             E(T(D)) ← E(T(D)) ∪ {(t, t')}
14.         retorna D
15. fim
```

Figura 4.1. Algoritmo para decomposição em árvore de grafos cordais.

Pelo Lema 2.9, todo separador minimal de um grafo cordal é uma clique. Logo, dado um separador S de um grafo cordal G, cada componente C de $G - S$ é também cordal, o que justifica a recursão na linha 9. Como a base da recursão ocorre nas linhas 2 e 3, pode-se ver claramente que toda parte da decomposição é uma clique. Logo, como a maior clique de G está exatamente contida em alguma parte da decomposição obtida, esta é mínima. Resta justificar a existência de t' na linha 10; isto é imediato a partir do Lema 2.10, já que um vértice v tal que $\{v\} \cup S$ induz uma clique em $G - S$ certamente existe, e logo também existe um nó t' em $T_v(D')$ contendo $\{v\} \cup S$.

Através do algoritmo DEA_CORDAL, pode-se perceber que todo grafo cordal pode ser obtido recursivamente pela colagem de grafos cordais ao longo de subgrafos completos. Tal característica pode ser extremamente vantajosa no projeto de algoritmos para alguns problemas de otimização, já que algumas características de cada parte de uma decomposição obtida pelo algoritmo DEA_CORDAL podem ser estendidas para todo o grafo, como, por exemplo, a não continência de algum menor K_r, para algum r inteiro constante. Tal fato motiva a definição de uma decomposição em árvore particular:

Definição 22 (Decomposição em árvore simplicial). Uma decomposição em árvore $D = (\mathcal{X} = \{X_i\}_{i \in I}, T = (I, F))$ de um grafo $G = (V, E)$ é dita *simplicial* se, além das restrições usuais de uma decomposição em árvore, esta satisfaz:

[P4S] dados nós $i, j \in I$ de T, o separador $X_i \cap X_j$ forma uma clique em G.

Logo, a decomposição obtida pelo algoritmo DEA_CORDAL é simplicial. De fato, não é difícil perceber que todo grafo admite uma decomposição em árvore simplicial — embora não necessariamente mínima, como no caso dos grafos cordais — bastando para tanto aplicar recursivamente o mesmo procedimento usando separadores clique até que somente partes irredutíveis sejam obtidas. Como Diestel (2000) observa, se todos os separadores clique usados forem minimais, então o conjunto de partes obtido será até mesmo único.

Além de motivar uma abordagem mais abrangente para todos os grafos, o estudo de grafos cordais pode ainda fornecer elementos para a obtenção de um *limitante superior* para a largura em árvore de grafos em geral. Basta perceber, com a ajuda do Lema 3.9, que a largura em árvore de um grafo é no máximo a largura em árvore de qualquer cordalização deste. Desta forma, o algoritmo DEA_CORDAL pode ser modificado para encontrar uma cordalização de um grafo G, e conseqüentemente, um limitante para $LA(G)$, como mostrado na Figura 4.2.

A corretude de DEA_CORDAL_LS segue diretamente da de DEA_CORDAL. Vale notar que, a rigor, a escolha de um vértice na linha 2 para construção de um separador S pode ser feita arbitrariamente. A escolha de um vértice de grau mínimo é apenas uma regra heurística para que as cliques na cordalização resultante sejam menores.

Em (Koster et al. 2001) são mostrados outros algoritmos para limitantes inferiores e superiores da largura em árvore de um grafo G, baseados no maior grau mínimo encontrado para todos os subgrafos de G, e em cordalizações de G, respectivamente.

▷ **Algoritmo** DEA_CORDAL_LS(G)
▷ **Entrada:** Um grafo $G = (V, E)$.
▷ **Saída:** Uma decomposição em árvore $D = (\mathcal{X}, T)$ de G.

1. **início**
2. **selecione** $v : d(v) = \delta(G), v \in V$
3. $S \leftarrow N_G(v) \cup \{v\}$
4. $G' \leftarrow G + K(S)$
5. **se** G' é completo **então**
6. **retorna** $(\{V\}, (\{t\}, \emptyset))$
7. **senão**
8. $D \leftarrow (\{S\}, (\{t_S\}, \emptyset))$
9. **para** cada $C \in$ COMPONENTES($G[V \setminus S]$) **faça**
10. $D' \leftarrow$ DEA_CORDAL_LS($G'[C \cup S]$)
11. **selecione** $t' : S' \subseteq X_{t'}, t' \in T(D'), X_{t'} \in \mathcal{X}(D')$
12. $\mathcal{X}(D) \leftarrow \mathcal{X}(D) \cup \mathcal{X}(D')$
13. $T(D) \leftarrow T(D) \cup T(D')$
14. $E(T(D)) \leftarrow E(T(D)) \cup \{(t, t')\}$
15. **retorna** D
16. **fim**

Figura 4.2. Algoritmo para decomposição em árvore baseado em cordalização.

4.3 DEA usando separadores fortes

Na seção anterior viu-se como a noção de separadores pode ser extremamente útil na determinação de uma decomposição em árvore de um grafo. A introdução de um novo conceito de separador permite a obtenção de uma decomposição em árvore de largura limitada.

Definição 23 (Separador forte). Um *separador forte* em um grafo $G = (V, E)$ é um conjunto X de vértices, $X \subseteq V$, tal que nenhuma componente de $G - X$ contem mais de $\frac{2}{3}|V(G - X)|$ vértices. De modo semelhante, um S-separador forte para algum $S \subseteq V$ é um conjunto de vértices X tal que nenhuma componente de $G - X$ contem mais de $\frac{2}{3}|S - X|$ vértices de S. A *ordem* de um separador é a cardinalidade de X.

Vale observar que a definição encontrada em Reed (1992) refere-se apenas a um *separador* como sendo um conjunto satisfazendo as condições da definição acima; de modo a evitar confusão com o conceito tradicional de separador, a definição aqui empregada foi modificada para separador forte.

A primeira relação entre separadores fortes e largura em árvore é estabelecida pelo seguinte lema:

Lema 4.1 (Reed [1992]). *Seja $G = (V, E)$ um grafo. Se G tem largura em árvore no máximo k, então para todo conjunto $S \subseteq V$, G contém um S-separador forte de ordem no máximo $k + 1$.*

Prova. Claramente, todo conjunto S com no máximo $k+1$ vértices admite a si mesmo como separador forte, e logo apenas conjuntos S com $|S| > k+1$ serão considerados. O objetivo da prova é mostrar que ou G tem um S-separador forte de ordem no máximo $k+1$ ou G tem um arbusto de ordem no mínimo $k+2$, e logo, pelo Teorema da dualidade, G tem largura em árvore no mínimo $k+1$.

Considere então a família $\mathcal{B}_S$ de subgrafos conexos de G tais que $B \in \mathcal{B}_S$ se e somente se $|N_G(B)| \leq k+1$ e $|B \cap S| > \frac{2}{3}|S - N_G(B)|$. Suponha agora que existam dois elementos B_1 e B_2 de $\mathcal{B}_S$ que não se tocam. Logo, como $B_1 \cap B_2 = \emptyset$ e não existem arestas entre B_1 e B_2, tem-se que $B_1 \subseteq G - N_G(B_1) - B_2 - N_G(B_2)$ e $B_2 \subseteq G - N_G(B_2) - B_1 - N_G(B_1)$. Daí segue que $(B_1 \cap S) \cup (B_2 \cap S) \subseteq S - N_G(B_1) - N_G(B_2)$, e logo $|B_1 \cap S| + |B_2 \cap S| \leq |S - N_G(B_1) - N_G(B_2)|$. Por outro lado, como $|B_1 \cap S| > \frac{2}{3}|S - N_G(B_1)|$ e $|B_2 \cap S| > \frac{2}{3}|S - N_G(B_2)|$ pela presença de ambos em $\mathcal{B}_S$, e como $|S - N_G(B_1)| + |S - N_G(B_2)| \geq 2 \cdot |S - N_G(B_1) - N_G(B_2)|$, tem-se que $|B_1 \cap S| + |B_2 \cap S| > \frac{4}{3}|S - N_G(B_1) - N_G(B_2)|$. Da contradição encontrada, conclui-se que quaisquer dois elementos de $\mathcal{B}_S$ se tocam, e logo $\mathcal{B}_S$ é um arbusto. Mas se um conjunto de acerto X para $\mathcal{B}_S$ tem ordem no máximo $k+1$ então nenhuma componente de $G - X$ contém mais de $\frac{2}{3}|S - X|$ vértices, caso contrário tal componente estaria em $\mathcal{S}$ — pois X seria a vizinhança desta componente — contradizendo o fato de X ser um conjunto de acerto. Logo, ou X é um S-separador forte ou $\mathcal{B}_S$ tem ordem no mínimo $k+2$, o que conclui a prova. □

A relação recíproca a do Lema 4.1 garante uma condição suficiente mais relaxada para a largura em árvore máxima de um grafo:

Lema 4.2 (Reed [1992]). *Seja $G = (V, E)$ um grafo. Se, para todo $S \subseteq V$, G contém um S-separador forte de ordem $k+1$, então G tem largura em árvore no máximo $4k+1$.*

Com base no Lema 4.2, pode-se ver que se um grafo G não possui S-separadores fortes, para todo $S \subseteq V$, então a largura em árvore de G é no máximo $4k+1$. A prova do Lema 4.2 é imediata se uma rotina recursiva é executada com base neste fato: basta encontrar um S-separador forte X e aplicar a rotina em cada componente de $G - X$. Como $|X| \leq k+1$, tem-se que $|S| \leq 3k+1$ — o que ficará mais claro após a prova do Lema 4.3 — e logo segue que $|X \cup S| \leq 4k+1$. O argumento deste esboço de prova fica mais claro se a rotina recursiva é posta em forma de um algoritmo, como proposto por Reed (1992):

1. Encontre um S-separador forte X de ordem no máximo $k+1$. Se tal separador forte não existe, então pare, e retorne S. Caso contrário, sejam $U_1, \ldots, U_l$ as componentes de $G - X$, $G_i = X \cup U_i$ e $S_i = (U_i \cap S) \cup X$, $1 \leq i \leq l$.
2. Como X é um S-separador forte, $|S| \leq 3k+1$, e como $|X| \leq k+1$, tem-se que $|S_i| \leq 3k+1$, $1 \leq i \leq l$, e logo pode-se aplicar a rotina recursivamente em cada G_i procurando um S_i-separador forte. Caso não seja encontrado um separador forte de ordem no máximo $k+1$ em algum G_i, pare: G também não contém um separador forte de ordem no máximo $k+1$.
3. Neste passo as rotinas recursivas para cada G_i foram bem sucedidas, retornando uma decomposição em árvore $D_i = (\mathcal{X}_i, T_i)$, onde existe um nó destacado t_i em

cada T_i tal que $S_i \subseteq X_{t_i}$. Uma decomposição em árvore $D = (\mathcal{X}, T)$ para G pode então ser encontrada com base nas D_i, onde:

(a) $V(T) = \left(\bigcup_{i=1}^{l} V(T_i)\right) \cup \{t\}$;

(b) $E(T) = \left(\bigcup_{i=1}^{l} E(T_i)\right) \cup \{(t, t_i) \,:\, 1 \leq i \leq l\}$;

(c) $\mathcal{X} = \left(\bigcup_{i=1}^{l} \mathcal{X}_i\right) \cup X_t$, $X_t = X \cup S$.

ou seja, D é uma decomposição formada pela união entre as l decomposições D_i pela adição de um nó t que é feito adjacente a cada nó destacado t_i, sendo a parte correspondente a t igual a união entre X e S, X_t (Figura 4.3). Retorne D.

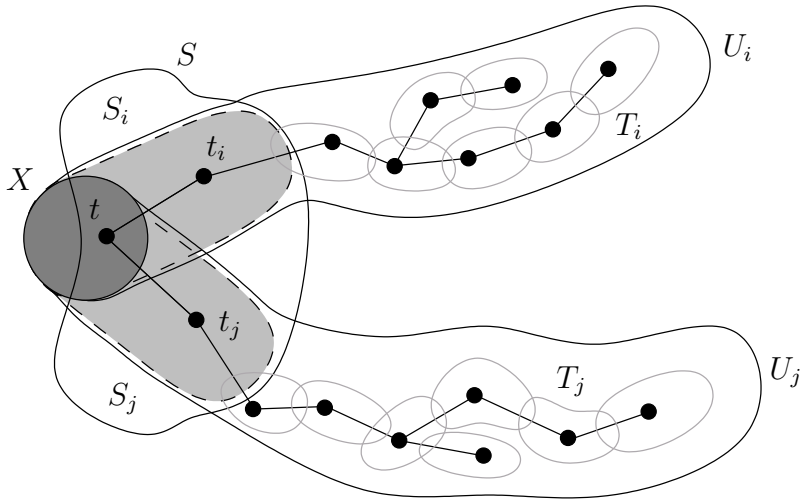


Figura 4.3. Decomposição em árvore obtida pelo algoritmo de Reed: os ramos correspondentes a duas componentes, U_i e U_j são mostrados, assim como os novos separadores fortes para cada componente, S_i e S_j, respectivamente, em cinza, e os nós destacados de cada árvore T_i e T_j.

Vale notar que a construção para obtenção de D no passo 3 já é familiar, sendo encontrada uma forma semelhante na prova da Propriedade 3. Não é difícil verificar que D é realmente uma decomposição em árvore, já que X_t sempre contém a interseção entre quaisquer duas partes correspondentes a dois nós destacados. Além disso, como a maior parte de uma decomposição D_i contém no máximo a união entre seu S_i-separador forte e S_i, segue que $LA(D_i) \leq 4k + 1$. De fato, a rotina apresentada acima é uma prova algorítmica do Lema 4.2. O algoritmo de Reed é formalizado na Figura 4.4

A análise de complexidade segue com a ajuda do seguinte lema:

Lema 4.3 (Reed [1997]). *Dado um conjunto $S \subseteq V$ de um grafo $G = (V, E)$, com $|S| \leq 3k + 1$, existe uma rotina k-*SEPARADOR *que retorna um S-separador forte de ordem no máximo $k + 1$, se este existe, com complexidade de tempo $O(k \cdot 3^{3k+1} \cdot |E|)$.*

▷ **Algoritmo** k-DEA_REED(G, S)
▷ **Entrada:** Um grafo $G = (V, E)$ e $S \subseteq V$ tal que $|S| \leq 3k + 1$.
▷ **Saída:** Se $LA(G) \leq k$, uma decomposição em árvore $D = (\mathcal{X}, T)$ de G com largura no máximo $4k + 1$; caso contrário, um subconjunto S de vértices tal que G não contém um S-separador forte de ordem no máximo $k + 1$.

1. **início**
2. **se** $|V| \leq 4k + 1$ **então**
3. **retorna** $(\{V\}, (\{t\}, \emptyset))$
4. **senão**
5. $X \leftarrow k$-SEPARADOR(S, G)
6. **se** $X = \emptyset$ **então**
7. **retorna** S
8. **senão**
9. $D \leftarrow (\{X \cup S\}, (\{t\}, \emptyset))$
10. **para** cada $C \in$ COMPONENTES$(G[V \setminus X])$ **faça**
11. $G' \leftarrow G[C \cup X]$
12. $S' \leftarrow (C \cap S) \cup X$
13. $D' \leftarrow k$-DEA_REED(G', S')
14. **se** D' é um conjunto de vértices **então**
15. **retorna** S'
16. **senão**
17. **selecione** $t' : S' \subseteq X_{t'}, t' \in T(D'), X_{t'} \in \mathcal{X}(D')$
18. $\mathcal{X}(D) \leftarrow \mathcal{X}(D) \cup \mathcal{X}(D')$
19. $T(D) \leftarrow T(D) \cup T(D')$
20. $E(T(D)) \leftarrow E(T(D)) \cup \{(t, t')\}$
21. **retorna** D
22. **fim**

Figura 4.4. Algoritmo proposto em Reed (1992).

Prova. Considere um conjunto S de vértices de um grafo $G = (V, E)$, $|S| \leq 3k + 1$, como no enunciado. Inicialmente, deve-se mostrar que G admite um S-separador forte se e somente se V pode ser particionado em 3 conjuntos A, B e X tais que cada componente de $G - X$ está contida em A ou B, $|X| \leq k + 1$, $|A \cap S| \leq \frac{2}{3}|S - X|$ e $|B \cap S| \leq \frac{2}{3}|S - X|$. A implicação contrária é imediata: basta tomar X como S-separador forte, e logo, como toda componente C de $G - X$ é tal que $|C| \leq \frac{2}{3}|S - X|$, e como $A \cap B = \emptyset$ já que (A, B, X) é uma partição de V, segue que $C \subseteq A$ ou $C \subseteq B$. Para o sentido inverso, seja X também o S-separador forte, e sejam $U_1, \ldots, U_l$ as componentes de $G - X$ ordenadas decrescentemente de acordo com a cardinalidade da interseção com S, ou seja, $|U_i \cap S| \geq |U_{i+1} \cap S|$. Basta agora definir $A = \bigcup_{i=1}^{j} U_j$, onde j é o menor índice tal que $|\bigcup_{i=1}^{j+1}(S \cap U_i)| \geq \frac{2}{3}|S - X|$; e $B = V \setminus A \setminus X$.

A idéia da prova agora é encontrar uma partição (A, B, X) de V de modo que X seja um S-separador forte. Para tanto, podem-se considerar as combinações de vértices

de S nos conjuntos de uma partição (S_A, S_B, S_X) — ou seja, S_A, S_B e S_X são disjuntos e $S_A = S \cap A$, $S_B = S$ — tal que $S_A = S \cap A$, $S_B = S \cap B$ e $S_X = S \cap X$. Desta forma, a idéia é encontrar uma partição em S e estendê-la de modo a obter uma partição correspondente em G contendo um S-separador forte.

Como para cada vértice em S existem três opções — estar em S_A, S_B ou S_X — e S tem no máximo $3k+1$, tem-se um total de no máximo 3^{3k+1} combinações possíveis para uma partição de S. Este total de combinações fica reduzido se as restrições para a partição em V forem aplicadas para S: como $|X| \leq k+1$, deve-se ter $|S_X| \leq k+1$, e como $|A \cap S| \leq \frac{2}{3}|S - X|$ e $|B \cap S| \leq \frac{2}{3}|S - X|$, deve-se ter também $|S_A| \leq 2|S_B|$ e $|S_B| \leq 2|S_A|$. Note agora que, assim como A e B são separados por no máximo $k+1$ caminhos disjuntos em G — já que X é um separador de ordem no máximo $k+1$, e portanto contém pelo menos um vértice de cada um destes caminhos — assim também S_A e S_B devem ser separados por, no máximo, $k+1-|S_X|$ caminhos disjuntos em $G-S_X$. Se existem mais de $k+1-|S_X|$ caminhos disjuntos entre S_A e S_B em $G - S_X$, então existem mais de $k+1$ caminhos disjuntos entre A e B em G e logo X não pode ser um separador forte com no máximo $k+1$ vértices. A Figura 4.5 facilita a compreensão da relação entre as partições.

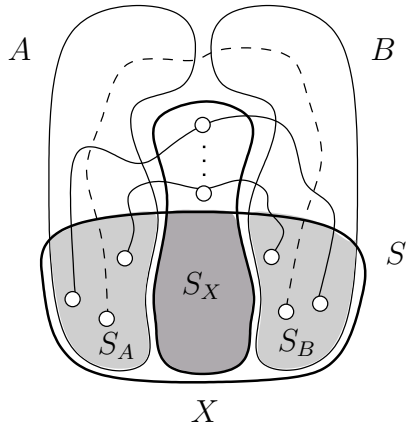


Figura 4.5. Construção de um S-separador forte de ordem no máximo $k+1$.

Suponha então que G admite um S-separador de ordem no máximo $k+1$, e logo S admite uma partição tal que existem no máximo $k+1-|S_X|$ caminhos disjuntos entre S_A e S_B em $G-S_X$. Tais caminhos podem ser encontrados, aplicando-se técnicas de caminhos alternados baseadas no Teorema de Menger, em tempo $O(k|E|)$. Seja $P = \{v_1, \ldots, v_r\}$, $r \leq k+1-|S_X|$, um conjunto contendo um vértice interno em cada caminho disjunto encontrado entre S_A e S_B. Basta ver agora que X pode então ser definido como $P \cup S_X$, A como a união das componentes de $G - X$ que contém pelo menos um vértice em S_A e $B = V \setminus A \setminus X$. Como para cada combinação de uma partição os caminhos disjuntos devem ser avaliados, a complexidade total da rotina é $O(k \cdot 3^{3k+1} \cdot |E|)$. □

Como pelo Lema 3.17 tem-se $E(G) < k|V(G)|$ para qualquer grafo G com largura em árvore no máximo k, segue que $O(|E(G)|) = O(|V(G)|)$, e logo a complexidade de

k-SEPARADOR é $O(|V|)$. Vale notar que, embora k seja uma constante, contribui com um grande fator exponencial que fica escondido na notação assintótica de complexidade.

Não é difícil perceber que, para cada passo da recursão, a rotina é executada para cada componente de $G - X$, ou seja, em $l = O(|V|)$ sub-problemas disjuntos, e logo a complexidade global de k-DEA_REED é $O(|V|^2)$. Acrescentando a noção de S-separador forte *aproximado*, Reed (1992) consegue dividir os sub-problemas de maneira que cada um tenha tamanho no máximo $(1-\epsilon)\cdot|V|)$, $\epsilon > 0$, e logo pode-se concluir que a complexidade após a modificação torna-se $O(|V|\log|V|)$. O preço deste ganho em desempenho é a obtenção de decomposições em árvore com largura no máximo $5k+1$, a partir de separadores fortes de ordem no máximo $4k+1$.

4.4 DEA em tempo linear

Um dos principais resultados para decomposições em árvore é a obtenção de um algoritmo linear para resolução de k-LARGURAEMÁRVORE com k constante. Tal resultado pode ser encontrado em (Bodlaender 1996), e resumido pelo seguinte teorema:

Teorema 4.4 (Bodlaender [1996]). *Para todo $k \in \mathbb{N}$, existe um algoritmo em tempo linear que testa se um dado grafo $G = (V, E)$ tem largura em árvore no máximo k, e, em caso positivo, retorna uma decomposição em árvore de G com largura em árvore no máximo k.*

A idéia principal do algoritmo é a *partição* dos vértices em dois conjuntos: vértices de *baixo grau* e vértices de *alto grau*. Pode ser mostrado que, para grafos com largura em árvore limitada por uma constante k, existem apenas "poucos" vértices de alto grau. Dois casos são então possíveis:

1. Um número "suficiente" de vértices de baixo grau é adjacente a um ou mais vértices de baixo grau. Neste caso, pode ser mostrado que qualquer emparelhamento maximal em G contém $\Omega(n)$ arestas. Computa-se então o grafo G' obtido pela contração de todas as arestas no emparelhamento maximal. Recursivamente, pode-se computar uma decomposição em árvore de largura no máximo k de G', ou concluir que a largura em árvore de G' é maior que k — e logo também a de G. Desta decomposição em árvore, pode-se facilmente construir uma decomposição em árvore de G com largura no máximo $2k+1$. Esta última decomposição é usada para resolver o problema, usando o algoritmo de Bodlaender e Kloks (1996).

2. Um número "pequeno" de vértice de baixo grau é adjacente a um ou mais vértices de baixo grau. Pode ser mostrado que uma certa quantidade de arestas pode ser adicionada a G sem tornar a sua largura em árvore maior que k, caso esta não seja maior que k. Pode ser mostrado que G', o *grafo melhorado de G* — G acrescido do novo conjunto de arestas — tem um número suficiente de vértices que são *I-simpliciais*: seus vizinhos formam uma clique no grafo melhorado, e algumas outras condições são satisfeitas. Recursivamente, uma decomposição em árvore de largura no máximo k é calculada de G', obtida pela remoção de todos os vértices I-simpliciais do grafo melhorado de G, ou pode-se concluir que a largura em árvore

de G', e logo também de G, é maior que k. A partir de tal decomposição em árvore de G', pode-se facilmente construir uma decomposição em árvore de G com largura no máximo k.

Alguns resultados auxiliares são necessários para compreensão do algoritmo.

Definição 24 (Decomposição em árvore suave). Uma decomposição em árvore $D = (\mathcal{X} = \{X_i : i \in I\}, T = (I, F))$ de largura k é dita *suave*, se para todo $i \in I$: $|X_i| = k + 1$, e para toda $(i, j) \in F : |X_i \cap X_j| = k$.

Bodlaender (1996) mostra que qualquer decomposição em árvore pode ser transformada numa decomposição suave de mesma largura, aplicando o seguinte conjunto de operações até que nenhuma seja mais possível:

- Se para $(i, j) \in F$, $X_i \subseteq X_j$, então contraia a aresta (i, j) em T e tome como novo nó $X_{j'} = X_j$;
- Se para $(i, j) \in F$, $X_i \not\subseteq X_j$ e $|X_j| < k + 1$, então escolha um vértice $v \in X_i \setminus X_j$, e adicione v a X_j;
- Se para $(i, j) \in F$, $|X_i| = |X_j| = k + 1$, e $|X_i \setminus X_j| > 1$, então subdivida a aresta (i, j) em T, e seja i' o novo nó. Escolha agora um vértice $v \in X_i \setminus X_j$ e um vértice $w \in X_j \setminus X_i$, e faça $X_{i'} = X_i \setminus \{v\} \cup \{w\}$.

Pelas propriedades de uma decomposição em árvore suave, pode-se estabelecer uma relação direta entre o número de vértices de um grafo, o número de partes de uma decomposição suave e a largura desta.

Lema 4.5. *Se $D = (\mathcal{X}, T)$ é uma decomposição em árvore suave de $G = (V, E)$ com largura k, então $|I| = |V| - k$.*

Prova. Por indução em $|I|$. □

Seja c_1 uma constante denotando o limite superior da fração de vértices que são de "alto" grau. Então uma outra constante c_2 pode ser definida por:

$$c_2 = \frac{1}{4k^2 + 12k + 16} - \frac{c_1 \cdot k^2(k + 1)}{2} > 0$$

As definições de vértices de baixo e alto grau podem agora ser estabelecidas:

Definição 25 (Vértices de baixo e alto grau). Seja $d = \max(k^2 + 4k + 4, \lceil 2k/c_1 \rceil)$. Um vértice v é dito *de baixo grau* se o grau de v é *no máximo* d, e é dito *de alto grau* se o grau de v é *maior* que d.

Definição 26 (Vértice amigável). Um vértice v é dito *amigável* se é de baixo grau e adjacente a pelo menos um outro vértice de baixo grau.

Primeira parte

A idéia da primeira parte do algoritmo é realizar um emparelhamento maximal nas arestas, contrai-las e então aplicar um novo algoritmo de decomposição com base no novo grafo obtido.

Inicialmente, deve-se garantir a condição de que se um número suficiente de vértices é de baixo grau, uma boa decomposição pode ser obtida a partir do emparelhamento. O primeiro resultado usa os vértices amigáveis do grafo.

Lema 4.6. *Se existem n_f vértices amigáveis em $G = (V, E)$, então qualquer emparelhamento maximal de G contém pelo menos $n_f/(2d)$ arestas.*

Prova. Considere um emparelhamento maximal M. Qualquer vértice amigável v deve ser uma extremidade de uma aresta em M, ou adjacente a um outro vértice amigável que é extremidade de uma aresta em M. Para cada aresta e em M, pode-se associar no máximo $2d$ vértices amigáveis que são extremidades de e ou adjacentes a um vértice amigável de e. Se um vértice amigável não está associado a alguma aresta em M, então M não é maximal. Conseqüentemente $|M| \geq n_f/(2d)$. □

A partir da contração das arestas em M, pode-se obter um novo grafo, e, a partir deste, proceder recursivamente na construção de uma decomposição em árvore. A relação entre as duas decomposições — a do grafo original e do obtido pela contração das arestas no emparelhamento maximal — é estabelecida por uma função de correspondência. Formalmente, sejam M um emparelhamento maximal em $G = (V, E)$ e $G' = (V', E')$ o grafo obtido pela contração das arestas de M em G. Define-se então $f_M : V \mapsto V'$ como $f_M(v) = v$ se v não é uma extremidade de uma aresta em M, e seja $f_M(v) = f_M(w)$ o vértice resultante da contração de $(v, w) \in M$. A decomposição em árvore de G pode então ser obtida a partir de G':

Lema 4.7. *Sejam M, G, G', f_M como acima. Se $(\mathcal{W}, T)$ é uma decomposição em árvore de G' com largura k, então $(\mathcal{X}, T)$, onde $X_i = \{v \in V : f_M(v) \in W_i\}$, é uma decomposição em árvore de G com largura no máximo $2k + 1$.*

Lema 4.8. *Sejam G e G' como definidos acima. A largura em árvore de G' é no máximo a largura em árvore de G.*

Prova. Basta observar que G' é um menor de G, e logo, pela Propriedade 4, $LA(G') \leq LA(G)$. □

Os Lemas 4.7 e 4.8 sugerem então uma boa estratégia: encontrar, inicialmente, uma decomposição em árvore D a partir da contração das arestas de um emparelhamento maximal e, em seguida, uma nova decomposição D' a partir da função f_M. Como a largura de D' é limitada — no máximo $2 \cdot LA(D) + 1$ — basta encontrar um algoritmo para encontrar uma nova decomposição D'' a partir de D', tendo esta largura no máximo $LA(D)$. Bodlaender e Kloks (1996) sugeriram tal algoritmo, encontrando o seguinte resultado:

Teorema 4.9. *Para todo k, l, existe um algoritmo em tempo linear que, dados um grafo $G = (V, E)$ e uma decomposição em árvore $(\mathcal{X}, T)$ de G com largura no máximo l, determina se a largura em árvore de G é no máximo k, e, caso positivo, encontra uma decomposição em árvore de G com largura no máximo k.*

Segunda parte

Na segunda parte do algoritmo, um número pequeno de vértices é de baixo grau; neste caso, novos conceitos são necessários. Assim como na primeira parte um novo grafo é obtido a partir do grafo original, também na segunda parte busca-se uma operação que simplifique a busca de uma decomposição em árvore limitada.

A idéia da primeira parte é aproveitar o fato de que a maioria dos vértices são de baixo grau e contrair arestas para obtenção de um novo grafo com largura em árvore limitada, controlada. A segunda parte, no entanto, implementa uma abordagem de certa forma contrária: se uma pequena parte dos vértices for de baixo grau, pode-se usar esta característica para acrescentar arestas e manter a largura em árvore do grafo, enquanto a busca de decomposições é facilitada. Considere então o seguinte conceito:

Definição 27 (Grafo melhorado). Dado um grafo $G = (V, E)$, o grafo $G' = (V, E')$ é dito *melhorado* de G, sendo obtido pela adição de arestas (v, w) a E para todos os pares $v, w \in V$ tais que v e w tem pelo menos $k + 1$ vizinhos em comum de baixo grau em G.

A relação entre G e seu grafo melhorado G' é mostrada no seguinte lema:

Lema 4.10. *Se a largura em árvore de G é no máximo k, então a largura em árvore do grafo melhorado de G é também no máximo k. Além disso, qualquer decomposição em árvore de G com largura no máximo k é também uma decomposição em árvore do grafo melhorado com largura no máximo k, e vice-versa.*

Prova. A prova é trivial, e segue diretamente da Propriedade 8. □

A partir do grafo melhorado de G pode-se acrescentar uma nova definição:

Definição 28 (Vértice I-simplicial). Um vértice de um grafo G é dito *I-simplicial* se é simplicial no grafo melhorado de G, de baixo grau e não amigável em G.

Os lemas que se seguem mostram que, se G tem poucos vértices de alto grau e poucos vértices amigáveis, então G tem muitos vértices I-simpliciais. Antes, no entanto, um outro conceito se faz necessário:

Definição 29 (Vértice útil). Um vértice v de um grafo G é dito *útil* com respeito a alguma decomposição em árvore $(\mathcal{X}, T)$ se é de baixo grau, não amigável, e existe um nó $i \in I$ tal que todos os viznhos de v pertencem a X_i.

O próximo resultado compõe a base do algoritmo da segunda parte. Os lemas restantes procuram apenas estabelecer uma abordagem algorítmica a partir da sua interpretação.

Lema 4.11. *Seja $(\mathcal{X}, T)$ uma decomposição em árvore suave de $G = (V, E)$ com largura em árvore k.*

1. *Para toda folha i de T, pode-se associar um vértice de baixo grau $v \in X_i$ tal que este seja amigável ou útil com respeito a $(\mathcal{X}, T)$, e não exista um $j \in I$, $j \neq i$, tal que $v \in X_j$.*
2. *Para cada caminho $\langle i_0, i_1, \ldots, i_{k^2+3k+3} \rangle$ em T com $i_1, \ldots, i_{k^2+3k+2}$ nós de grau 2 em T, pode-se associar pelo menos um vértice $v \in X_{i_1} \cup \cdots \cup X_{i_{k^2+3k+2}}$ que seja*

amigável ou útil com respeito a $(\mathcal{X}, T)$, e tal que v não pertença a um conjunto X_j com $j \in I$ um nó não pertencente ao caminho.

Prova.

1. Seja j o vizinho da folha i em T. Seja v o único vértice em $X_i \setminus X_j$, e logo v é adjacente somente a vértices em X_i. Dois casos podem ocorrer: ou todos os vizinhos de v são de alto grau, onde neste caso v é útil com respeito a $(\mathcal{X}, T)$; ou um vizinho de v é de baixo grau, e neste caso v é amigável.
2. Note que $|X_{i_0} \cup \cdots \cup X_{i_{k^2+3k+3}}| = k^2 + 4k + 4 \leq d$. Conseqüentemente, todos os vértices em $X_{i_1} \cup \cdots \cup X_{i_{k^2+3k+2}} \setminus (X_{i_0} \cup X_{i_{k^2+3k+3}})$ são de baixo grau. Suponha que nenhum deles é amigável, isto é, eles são adjacentes somente a vértices de alto grau em $X_{i_0} \cup X_{i_{k^2+3k+3}}$. Suponha que X_{i_0} contém r vértices de alto grau, $w_1, \ldots, w_r$. Claramente $r \leq |X_{i_0}| = k + 1$. Cada um destes w_s pertencem a conjuntos sucessivos $X_{i_0}, X_{i_1}, \ldots, X_{i_{w_s}}$. Suponha, sem perda de generalidade, $i_{w_1} \leq i_{w_2} \leq \cdots \leq i_{w_r}$. Se algum vértice v de baixo grau pertence a exatamente um conjunto X_{i_j}, $1 \leq j \leq k^2 + 3k + 2$, então este deve ser útil com respeito a $(\mathcal{X}, T)$. Se algum vértice v de baixo grau pertence apenas aos conjuntos $X_{i_{w_j+1}}, \ldots, X_{i_{w_{j+1}}}$ (ou aos subconjuntos destes), então todos os vizinhos de v pertencem a $X_{i_{w_{j+1}}}$, e logo v é útil com respeito a $(\mathcal{X}, T)$. Todos os vértices em $X_{i_1} \cup \cdots \cup X_{i_{k^2+3k+2}}$ que não sejam de nenhum destes dois tipos devem pertencer a pelo menos um dos conjuntos $X_{i_0}, X_{i_{w_1}}, \ldots, X_{i_{w_r}}, X_{i_{k^2+3k+3}}$. Estes são no total no máximo $(k+1)(k+3) = k^2 + 4k + 3$ vértices. Logo, pelo menos um vértice em $X_{i_1} \cup \cdots \cup X_{i_{k^2+3k+2}} \setminus (X_{i_0} \cup X_{i_{k^2+3k+3}})$ deve ser útil com respeito a $(\mathcal{X}, T)$.

□

Definição 30 (Coleção folha-caminho). Uma *coleção folha-caminho* de uma árvore T é uma coleção de folhas em T, mais uma coleção de caminhos de comprimento k^2+3k+3 em T, onde todos os nós em um caminho que não sejam extremidades do caminho têm grau 2 em T e não pertencem a nenhum outro caminho na coleção. O *tamanho* da coleção é o número total de folhas mais o número total de caminhos na coleção.

Lema 4.12. *Cada árvore com r nós contém uma coleção folha-caminho de tamanho no mínimo $r/(2k^2 + 6k + 8)$.*

Prova. Sejam r_b o número de nós com grau no mínimo 3, r_f o número de folhas, e r_2 o número de nós com grau 2. Claramente, $r_b < r_f$. Todos os nós de grau 2 pertencem a menos de $r_f + r_b$ componentes conectadas da floresta obtida pela remoção de todas as folhas e todos os nós com grau 3 ou maior da árvore. Cada componente contém no máximo $k^2 + 3k + 3$ nós que não são parte de uma coleção folha-caminho de tamanho máximo. Logo, existem menos que $(r_f + r_b)(k^2 + 3k + 3)$ nós de grau 2 que não estejam em algum caminho da coleção. Conseqüentemente, existem no máximo

$$\frac{r_2 - (r_b + r_f)(k^2 + 3k + 3)}{k^2 + 3k + 4}$$

caminhos numa coleção folha-caminho de tamanho máximo. Segue que o tamanho máximo de uma coleção folha-caminho é no mínimo

$$\max\left(r_f, \frac{r_2 - (r_b + r_f)(k^2+3k+3)}{k^2+3k+4} + r_f\right) \geq \frac{1}{2} \cdot \frac{r}{k^2+3k+4}$$

□

Corolário 4.13. *Se $(\mathcal{X}, T)$ é uma decomposição em árvore suave de $G = (V, E)$ com largura k, então G contém pelo menos $|V|/(2k^2+6k+8)-1$ vértices que são amigáveis ou úteis com respeito a $(\mathcal{X}, T)$.*

Prova. T contém $|V| - k$ nós. Agora basta aplicar os dois últimos lemas, 4.11 e 4.12. □

Definição 31 (Conjunto semi-importante e importante). Um conjunto $Y \subseteq V$ de vértices de alto grau é dito *semi-importante* com respeito a uma decomposição em árvore $(\mathcal{X}, T)$ de $G = (V, E)$, se existe um $i \in I$ com $Y \subseteq X_i$. Um conjunto Y é dito *importante* se ele é semi-importante maximal com respeito a $(\mathcal{X}, T)$, ou seja, não está contido em nenhum conjunto semi-importante maior com respeito a $(\mathcal{X}, T)$.

Lema 4.14. *Seja $(\mathcal{X}, T)$ uma decomposição em árvore de $G = (V, E)$ com largura k. O número de conjuntos importantes diferentes com respeito a $(\mathcal{X}, T)$ é no máximo o número de vértices de alto grau em G.*

Prova. Seja L o conjunto de vértices de alto grau em G. $(\{X_i \cap L : i \in I\}, T)$ é uma decomposição em árvore de $G[L]$. Cada conjunto importante Y é um conjunto $X_i \cap L$ que não está contido em outro conjunto $X_{i'} \cap L$. A partir da contração repetitiva das arestas (i, i') em T com $X_i \cap L \supseteq X_{i'} \cap L$, de modo que o novo nó formado contenha todos os vértices em X_i, obtém-se uma nova decomposição. Tal decomposição em árvore resultante de $G[L]$ contém os mesmos conjuntos maximais X_i e terá no máximo $|L|$ nós. □

Definição 32 (Função UI). Uma função f que mapeia cada vértice útil v (com respeito a alguma decomposição em árvore $(\mathcal{X}, T)$) a um conjunto importante Y (também com respeito a $(\mathcal{X}, T)$) com $N_G(v) \subseteq Y$, é chamada *função UI* para $(\mathcal{X}, T)$. Por definição, uma função UI sempre existe.

Lema 4.15. *Seja f uma função UI para uma decomposição em árvore suave $(\mathcal{X}, T)$ de $G = (V, E)$ com largura k. Seja Y um conjunto importante com respeito a $(\mathcal{X}, T)$. Então no máximo $\frac{1}{2}k^2(k+1)$ vértices úteis com respeito a $(\mathcal{X}, T)$ em $f^{-1}(Y)$ não são I-simpliciais.*

Prova. Atribua, inicialmente, cada vértice útil v que não seja I-simplicial a um par de vizinhos de v que não sejam adjacentes no grafo melhorado de G. Pode-se ver que não se pode atribuir mais de k vértices para cada par de vértices em G, porque, caso contrário, estes teriam pelo menos $k+1$ vizinhos de baixo grau em comum, o que acrescentaria uma

aresta entre eles no grafo melhorado. Segue então que o número de vértices úteis e não I-simpliciais v com $f(v) = Y$ é no máximo $\frac{1}{2}|Y|(|Y|+1) \leq \frac{1}{2}k^2(k+1)$. □

Corolário 4.16. *Se $G = (V, E)$ tem largura em árvore no máximo k, e contém n_f vértices amigáveis e n_a vértices de alto grau, então existem pelo menos*

$$\frac{|V|}{2k^2+6k+8} - 1 - n_f - \frac{1}{2}k^2(k+1)n_a$$

vértices I-simpliciais em G.

Lema 4.17. *Seja $(\mathcal{X}, T)$ uma decomposição em árvore de largura no máximo k do grafo G' obtido pela remoção de todos os vértices I-simpliciais do grafo melhorado de $G = (V, E)$. Para todo vértice I-simplicial v, existe um $i \in I$ com $N_G(v) \subseteq X_i$.*

Prova. Basta ver que, por definição, vértices I-simpliciais não são adjacentes em G, e que sua vizinhança forma uma clique no grafo melhorado de G. Logo, basta aplicar a Propriedade 7 para concluir a prova. □

Algoritmo principal

O algoritmo apresentado por Bodlaender (1996) é recursivo, tendo como entrada um grafo G e uma constante k e como saída uma decomposição em árvore de G com largura no máximo k, se possível.

A base da recursão, ou seja o resultado para grafos com no máximo um número constante de vértices, pode ser implementada a partir de qualquer outro algoritmo força bruta, por exemplo. Caso G seja suficientemente grande, o algoritmo principal pode ser aplicado:

1. Verifique se $|E| \leq k|V| - \frac{1}{2}k(k+1)$. Caso contrário, pelo Lema 3.17, G tem largura em árvore maior que k: pare.
2. Conte o número de vértices amigáveis n_f em G. Se existem pelo menos $|V|/(4k^2 + 12k + 16)$ vértices amigáveis, então o algoritmo segue o primeiro caso:
   (a) Encontre um emparelhamento maximal M em G.
   (b) Obtenha o grafo G' pela contração das arestas em M de G.
   (c) Aplique o algoritmo recursivamente a G'.
   (d) Se G' tem largura em árvore maior que k, então pare, já que, pelo Lema 4.8, a largura em árvore de G deve também ser maior que k.
   (e) Suponha que a chamada recursiva tenha retornado uma decomposição em árvore $D' = (\mathcal{W}, T)$ de G' com largura em árvore k. Construa, com auxílio do Lema 4.7, uma nova decomposição $D^* = (\mathcal{X}, T)$ de G com largura no máximo $2k + 1$.
   (f) Use o algoritmo proposto no Teorema 4.9 para encontrar uma nova decomposição D a partir de D^*, k e $l = LA(D^*) \leq 2k + 1$.

3. Neste passo, existem menos de $|V|/(4k^2 + 12k + 16)$ vértices amigáveis, e logo deve-se seguir o segundo caso:
   (a) Encontre o grafo melhorado de G, G^*.
   (b) Se existe um vértice I-simplicial de grau no mínimo $k + 1$ então pare: G^* contém uma clique de tamanho $k + 2$, e logo, pelo Lema 4.10, a largura em árvore de G é também maior que k.
   (c) Ponha todos os vértices I-simpliciais no conjunto S. Encontre o grafo G' obtido pela remoção de todos os vértices I-simpliciais (em S) de G.
   (d) Se $|S| < c_2|V|$, então pare, já que a largura em árvore de G é maior que k. Pode-se chegar a esta conclusão a partir da definição de c_1 e do Corolário 4.16, onde, já que existem menos de $|V|/(4k^2+12k+16)$ vértices amigáveis, deve-se ter pelo menos $|V|/(4k^2+12k+16)-\frac{1}{2}k^2(k+1)c_1|V|$ vértices I-simpliciais em G^* — supondo que a largura em árvore de G seja no máximo k.
   (e) Como $|S| \geq c_2|V|$, aplique o algoritmo recursivamente a G'.
   (f) Se a largura em árvore de G' é maior que k, então pare, já que, como G' é um subgrafo de G, $LA(G) \geq LA(G')$ (Propriedade 2).
   (g) Suponha que a chamada recursiva tenha retornado uma decomposição em árvore $D' = (\mathcal{W}, T')$ de G' com largura em árvore k. Encontre uma nova decomposição em árvore $D = (\mathcal{X}, T')$ de G da seguinte forma:
      i. Inicialmente, faça D uma cópia de D', ou seja, acrescente a $\mathcal{X}$ todas as partes em $\mathcal{W}$, com $T = T'$.
      ii. Para cada vértice v em S, encontre $i_v \in I$ tal que $N_G(v) \subseteq X_{i_v}$. Pelo Lema 4.17, i_v sempre existe.
      iii. Para cada i_v, adicione um novo nó j_v a T adjacente a i_v, e uma parte $X_{j_v} = \{v\} \cup N_G(v)$ a $\mathcal{X}$.
4. Retorne D, uma decomposição em árvore de largura no máximo k.

Teorema 4.18. *O algoritmo proposto por Bodlaender (1996) tem complexidade de tempo linear.*

Prova. O tempo de execução do algoritmo pode ser estimado da seguinte forma: como um dos casos deve ser executado a partir da decisão no passo 2, o algoritmo executa recursivamente em um grafo $G = (V, E)$ com $(1-1/(2d(4k^2+12k+16))) \cdot |V|$ vértices — primeiro caso, descrito ao longo do passo 2 — ou em um grafo com $(1 - c_2) \cdot |V|$ vértices. Logo, seja $0 \leq c_3 < 1$ uma constante:

$$c_3 = \max\left(1 - c_2, 1 - \frac{1}{2d \cdot (4k^2 + 12k + 16)}\right)$$

Como todos os passos não recursivos executam em tempo linear, tem-se que, se o algoritmo toma tempo $T(n)$ em um grafo com n vértices no pior caso, então $T(n) \leq T(c_3 \cdot n) + O(n)$, e logo, usando técnicas convencionais de análise de complexidade, conclui-se que $T(n) = O(n)$. □

4.5 DEA usando redução

Enquanto que as abordagens apresentadas até agora baseiam-se em técnicas "tradicionais" de algoritmos em grafos — pesquisa de caminhos, separadores, busca em vizinhança e adição de arestas, por exemplo — a técnica de *redução* baseia-se fortemente em álgebra de grafos, como definida em (Arnborg et al. 1993), e em Lógica Monádica de Segunda Ordem, como em (Courcelle 1990). Uma revisão dos conceitos básicos na Seção 2.8 é recomendada para maior familiaridade com os novos conceitos a seguir introduzidos. A primeira definição merece um destaque por se tratar da base da técnica:

Definição 33 (Regra de redução). Uma *regra de redução* r é um par ordenado (H_1, H_2), onde H_1 e H_2 são grafos l-terminais, $l \geq 0$.

Um *encaixe* para uma regra de redução $r = (H_1, H_2)$ em um grafo G é um grafo l-terminal G_1 isomorfo a H_1 e tal que existe um outro grafo l-terminal G_2 com $G = G_1 \oplus G_2$ para algum $l \geq 0$. Se G contém um encaixe G_1 para r, então uma *aplicação* de r em G usando G_1 é uma operação que mapeia G a um grafo G' tal que se $G = G_1 \oplus G_3$, com G_1 isomorfo a H_1, então $G' = G_2 \oplus G_3$, com G_2 isomorfo a H_2. Diz-se também que G_1 foi *trocado* por G_2 em G, obtendo-se G'.

A aplicação de uma regra de redução pode ser simplesmente chamada de *redução*. A aplicação de r em G usando G_1 é também chamada de redução correspondente ao encaixe G_1, sendo denotada por $G \xrightarrow[G_1]{r} G'$. De um modo geral, as reduções em um grafo são executadas para algum encaixe, sem uma especificação deste, e logo denota-se a redução de G por r obtendo-se G' simplesmente por $G \xrightarrow{r} G'$. Obviamente, grafos diferentes podem ser obtidos a partir de G por redução por r, dependendo dos encaixes escolhidos. Se $\mathcal{R}$ é um conjunto $\{r_1, \ldots, r_s\}$ de regras de redução, então denota-se por $G \xrightarrow{\mathcal{R}} G'$ a redução de G por alguma regra $r \in \mathcal{R}$ obtendo-se G', ou seja, $G \xrightarrow{r} G'$ para algum $r \in \mathcal{R}$. Dado um conjunto de regras de redução $\mathcal{R}$, diz-se que G é *redutível* para $\mathcal{R}$ se e somente se existe um encaixe G_1 para alguma regra $r \in \mathcal{R}$ em G; caso contrário, diz-se que G é *irredutível* para $\mathcal{R}$.

A Figura 4.6 (a) ilustra a aplicação das regras de redução do conjunto $\mathcal{R} = \{r, r'\}$. Normalmente representa-se graficamente cada regra de redução por seus grafos terminais unidos por uma seta. Neste caso, se $r = (H_1, H_2)$, diz-se que H_1 é o *lado esquerdo* de r, enquanto que H_2 é o lado *direito* de r. Na Figura 4.6(b), G' é obtido da redução de G por r, $G \xrightarrow{r} G'$, enquanto G'' é uma redução de G' por r', e logo $G \xrightarrow{r} G' \xrightarrow{r'} G''$. Como ambas as regras r e r' de $\mathcal{R}$ não podem mais ser aplicadas a G'', este é claramente irredutível para $\mathcal{R}$.

De modo a se obter uma caracterização de uma propriedade de grafos por redução, deve-se levar em consideração as seguintes definições para conjuntos de regras de redução:

Definição 34. Seja P uma propriedade de grafos e $\mathcal{R}$ um conjunto de regras de redução.

- $\mathcal{R}$ é *seguro* para P se, sempre que $G \xrightarrow{\mathcal{R}} G'$, então $P(G) \Leftrightarrow P(G')$.

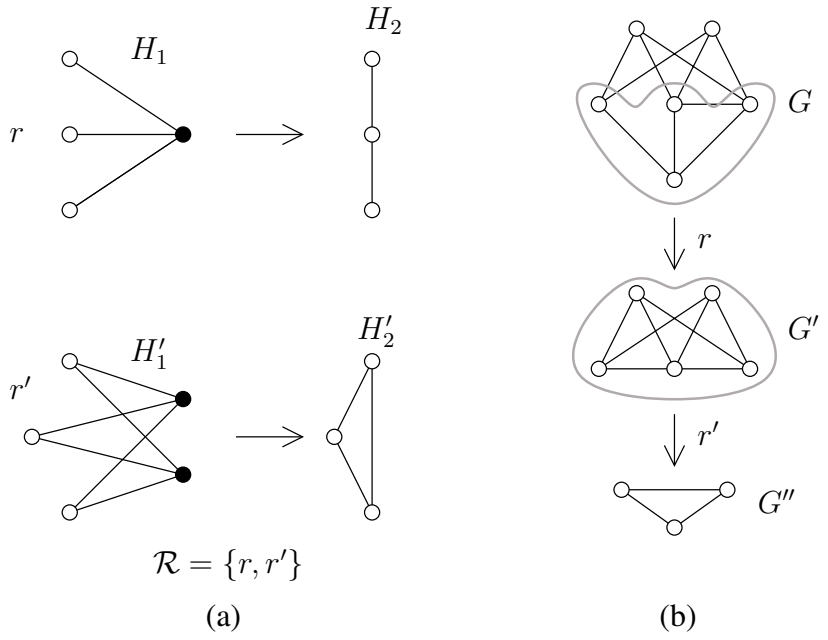


Figura 4.6. Exemplo de aplicação de regras de redução.

- $\mathcal{R}$ é *completo* para P se o conjunto I de grafos irredutíveis para $\mathcal{R}$ para os quais P vale é finito.
- $\mathcal{R}$ é *convergente* se não existe uma seqüência infinita $G_1 \xrightarrow{\mathcal{R}} G_2 \xrightarrow{\mathcal{R}} \cdots$.
- $\mathcal{R}$ é *decrescente* se, sempre que $G \xrightarrow{\mathcal{R}} G'$, então G' contém menos vértices que G.

Definição 35 (Sistema de redução). Um *sistema de redução* para uma propriedade de grafos P é um par $(\mathcal{R}, I)$, onde $\mathcal{R}$ é um conjunto finito de regras de redução seguro, completo e convergente para P, e I é o conjunto de grafos irredutíveis para $\mathcal{R}$ tal que $P(G)$ vale para todo $G \in I$, ou seja, $I = \{G : P(G) \wedge \neg\exists G' : G \xrightarrow{\mathcal{R}} G'\}$. Diz-se ainda que um sistema de redução $(\mathcal{R}, I)$ para P é *decrescente* se $\mathcal{R}$ é decrescente.

O conceito de sistema de redução decrescente sugere prontamente um algoritmo para caracterização de uma propriedade de grafos. Considere $(\mathcal{R}, I)$ um tal sistema, P uma propriedade e G um grafo a ser avaliado com relação a P: aplique as regras em $\mathcal{R}$ em G até que seja obtido um grafo G' irredutível; verifique se G' pertence a I e então decida pela satisfação de P por G, ou seja, em caso afirmativo, $G' \in I$ e logo G satisfaz P, caso contrário, P não vale para G.

O primeiro problema desta abordagem está na definição do sistema de redução, que, de modo geral, não é de fácil obtenção. Como exemplo, para uma caracterização dos grafos que são k-árvore parciais, com $0 \leq k \leq 3$ — e logo de largura em árvore no máximo k — Arnborg et al. (1993) sugerem conjuntos $(\mathcal{R}_k, I)$ seguros, completos, convergentes e decrescentes contendo as regras de redução mostradas na Figura 4.7. Nos quatro sistemas, os conjuntos de grafos irredutíveis são iguais e contém somente o grafo vazio, G_{vazio}. Os conjuntos de regras de redução são os seguintes: $\mathcal{R}_0 = \{r_1\}$, $\mathcal{R}_1 = \{r_1, r_2\}$, $\mathcal{R}_2 = \{r_1, r_2, r_3, r_4\}$ e $\mathcal{R}_3 = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$.

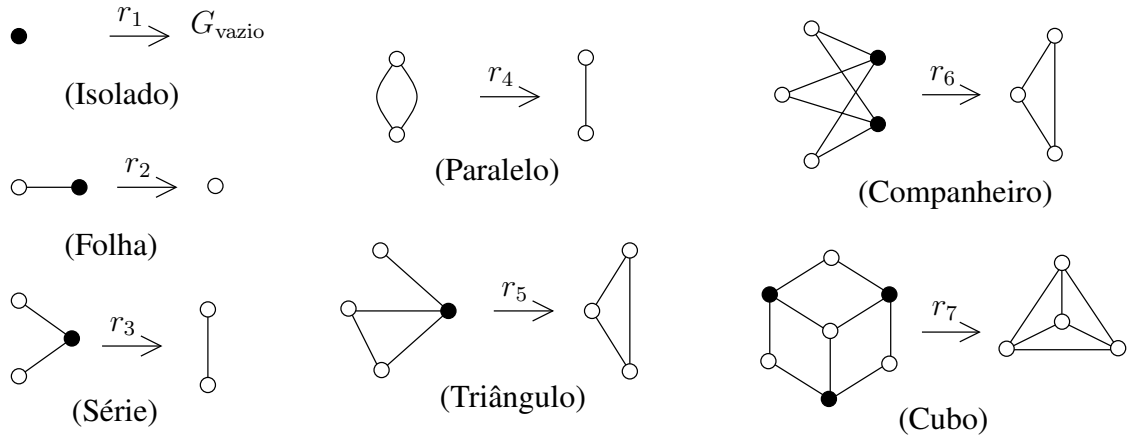


Figura 4.7. Regras de redução para conjuntos seguros, completos, convergentes e decrescentes para propriedade de ser uma k-árvore parcial, $0 \leq k \leq 3$.

Como resultado mais importante, Arnborg et al. (1993) mostraram que, para cada propriedade de grafos de índice finito e para cada k inteiro, existe um sistema de redução seguro, completo e convergente para os grafos com largura em árvore no máximo k. Em (de Fluiter e Bodlaender 1995) é encontrada uma lista de problemas relacionados à propriedades de índice finito em grafos com largura em árvore limitada. Em (Arnborg et al. 1991) é mostrado que o problema de determinar se a largura em árvore de um grafo é no máximo k, para k constante, refere-se a uma propriedade MS-definível, e logo de índice finito.

Supondo então que um sistema de redução seja conhecido para uma dada propriedade, o problema agora está em encontrar um encaixe para as regras do sistema de redução no grafo G de entrada. Uma rotina força bruta pode fazer uma verificação para conjunto de c vértices em G, onde c é o tamanho do maior grafo terminal nos lados esquerdos das regras, tendo portanto complexidade em tempo $O(n^c)$, $n = |V(G)|$.

Usando o fato de que propriedades de grafos de índice finito incluem todas as propriedades que podem ser expressas em LMSO, Arnborg et al. (1993) apresentam um algoritmo em tempo linear, mas de espaço polinomial $O(n^p)$ — p sendo o maior número de terminais dentre todos os lados esquerdos das regras — para decidir se uma dada propriedade P vale para um grafo G com largura em árvore limitada. O algoritmo de Arnborg et al. não depende da estrutura das regras de redução do sistema, podendo ser aplicado em qualquer conjunto de regras que seja finito, seguro, completo e decrescente (e logo, convergente).

Uma maneira de tornar um algoritmo mais eficiente é encontrar um conjunto de regras que tenha uma estrutura especial, de modo que o algoritmo possa tomar vantagem de tal estrutura sob a forma de ganho de desempenho e espaço em memória ao determinar se alguma regra de redução do sistema deve ser aplicada, e qual. Em (Bodlaender e Hagerup 1995) é apresentado um algoritmo paralelo baseado nesta idéia, usando um método chamado *busca limitada em listas de adjacência*. de Fluiter e Bodlaender (1995) adaptam uma versão seqüencial deste algoritmo, estendendo os resultados em Arnborg et al. (1993) de duas formas: encontrando um algoritmo construtivo para resolver o problema

de decisão associado à propriedade, e um método para resolução de certos problemas de otimização em grafos com largura em árvore limitada. Além disso, é mostrado que uma combinação destes resultados é ainda possível, resultando portanto numa versão construtiva de um algoritmo para resolução de certos problemas de otimização para grafos com largura em árvore limitada.

A discussão destes últimos algoritmos envolve conceitos mais técnicos, tanto de LMSO quanto de álgebra de grafos, e não será realizada aqui, dadas as limitações de escopo e nível pretendido do texto. No entanto, a versão inicial de decisão de uma propriedade será abordada, sendo apresentado o algoritmo de redução encontrado em (de Fluiter 1997). Mais conceitos são necessários:

Definição 36 (Grafo terminal d-descobrível). Sejam d um inteiro positivo, G um grafo dado, representado por listas de adjacência, e G_1 um grafo l-terminal. Diz-se então que G_1 é *d-descobrível* em G se

1. G_1 é aberto e conexo, e o máximo grau de qualquer vértice de G_1 é no máximo d, ou seja, $\Delta(G_1) \leq d$;
2. existe um grafo l-terminal G_2 tal que $G = G_1 \oplus G_2$;
3. G_1 contém um vértice interno v tal que, para todos os vértices $w \in V(G_1)$, existe uma trilha W em G_1 com $W = \langle u_1, \ldots, u_s \rangle$, com $u_1 = v$ e $u_s = w$, e, para cada i, $1 < i < s$, na lista de adjacência de u_i em G, as arestas (u_{i-1}, u_i) e (u_i, u_{i+1}) têm distância no máximo d.

Um candidato a encaixe d-descobrível possui estrutura especial que lhe garante uma vantagem algorítmica:

Lema 4.19 (de Fluiter [1997]). *Se um grafo terminal G_1 é d-descobrível em um grafo G para algum $d \geq 1$, então, para qualquer vértice interno de G_1, todos os vértices e arestas de G_1 podem ser encontrados a partir de v em complexidade de tempo que depende apenas de d e do tamanho de G_1, mas não do tamanho do grafo G.*

Prova. Sejam G um grafo, d um inteiro positivo, G_1 um grafo terminal d-descobrível em G, e v um vértice interno qualquer de G_1. Para provar o lema, basta mostrar que, para qualquer outro vértice w de G_1, pode-se encontrar uma trilha W satisfazendo a condição 3 da Definição 36 para G_1 e G e tal que o tamanho de W dependa apenas de d e G_1, e logo qualquer vértice de G_1 pode ser encontrado em tempo também dependente apenas de d e G_1.

Se W é uma trilha como acima, é sempre possível encontrar uma trilha W' derivada de W, e logo também entre v e w em G_1, tal que qualquer aresta $e = (x, y)$ aparece no máximo duas vezes em W'. Suponha então que exista uma aresta $e = (x, y)$ que aparece no mínimo três vezes em W; logo, como e aparece pelo menos duas vezes em W no mesmo sentido, W contém uma sub-trilha W'' da forma $\langle x, y, \ldots, x, y \rangle$ (ou $\langle y, x, \ldots, y, x \rangle$) e W' pode ser obtida pela remoção da sub-trilha $\langle y, \ldots, x \rangle$ (ou $\langle x, \ldots, y \rangle$) em W'' de W. Não é difícil perceber que $W' = \langle u'_1, \ldots, u'_p \rangle$ é tal que $u'_1 = v$, $u'_p = w$ e, para cada i, $1 < i < p$, as arestas (u'_{i-1}, u'_i) e (u'_i, u'_{i+1}) têm distância no máximo d na lista de adjacência de G, e logo, W' também satisfaz a condição 3.

Desta forma, sempre pode-se obter uma trilha W entre um vértice interno v e w em G_1 tal que qualquer aresta em W apareça no máximo duas vezes. O resultado do lema — W poder ser encontrada em tempo dependente apenas de d e G_1 — segue agora do fato de que G_1 é aberto, e logo todo terminal é adjacente a um vértice interno. □

Considere agora a seguinte definição:

Definição 37 (Sistema de redução especial). Sejam P uma propriedade de grafos e $(\mathcal{R}, I)$ um sistema de redução decrescente para P. Seja $n_{\max}$ o número máximo de vértices em qualquer lado esquerdo das regras em $\mathcal{R}$. $(\mathcal{R}, I)$ é um *sistema de redução especial* para P se existem inteiros positivos $n_{\min}$ e d, $n_{\min} \leq n_{\max} \leq d$, tais que as seguintes condições valem:

1. Para toda regra de redução $(H_1, H_2) \in \mathcal{R}$:
   (a) se H_1 tem pelo menos um terminal, então H_1 é conexo e H_1 e H_2 são abertos;
   (b) se H_1 é um grafo zero-terminal, então $|V(H_2)| < n_{\min}$.
2. Para todos os grafos G tais que $P(G)$ vale, e todas as representações de G em lista de adjacências:
   (a) cada componente de G com pelo menos $n_{\min}$ vértices tem um encaixe d-descobrível;
   (b) se todas as componentes de G têm menos que $n_{\min}$, então ou $G \in I$ ou G contém um encaixe que é um grafo zero-terminal.

As condições 1(a) e 2(a) garantem que cada componente H de um grafo G, com $|V(H)| \geq n_{\min}$ e $P(G)$ valendo, contém pelo menos um encaixe d-descobrível para uma regra de redução, e este pode ser aplicado sem remoção de aresta múltiplas, já que ambos os lados da regra são abertos. As condições 1(b) e 2(b), por sua vez, são somente necessárias para propriedades de grafos que não consideram o grafo de entrada como conexo, e garantem que, se nenhuma outra regra de redução é aplicável, então encaixes para regras com lado esquerdo contendo um grafo zero-terminal e desconexo podem ser encontrados eficientemente.

Com base na definição de sistema de redução especial, pode-se projetar um algoritmo em tempo e espaço lineares. O algoritmo consiste de duas fases, ilustradas nas Figuras 4.8 e 4.9, respectivamente, cada uma tomando vantagem de cada condição do sistema de redução especial.

Na primeira fase, o algoritmo encontra encaixes d-descobríveis (linha 6) e executa as reduções correspondentes (linhas 8 a 8) até que o grafo de entrada G torne-se irredutível com relação a encaixes d-descobríveis, ou seja, com relação a regras de redução com lados esquerdos possuindo terminais. A irredutibilidade de G é verificada com base no conjunto S de vértices de grau no máximo d, sendo este atualizado a partir das listas de adjacência de G, no laço da linha 18. Vale notar que a função d-DESCOBRÍVEL, que tem como objetivo encontrar um encaixe d-descobrível G_1 dentre as regras de $\mathcal{R}$ tal que v é interno em G_1, retorna r e tem complexidade de tempo estabelecida pelo Lema 4.19.

Na linha 23, G pode ser testado para pertinência em I, e o algoritmo retorna verdadeiro em caso positivo. Caso contrário, o algoritmo segue para a segunda fase, na

```
▷ Algoritmo REDUÇÃO(G, R, I) — Fase 1
▷ Entrada: Um grafo G = (V, E) e um sistema de redução especial (R, I)
   para uma propriedade de grafos P e com parâmetros n_min e d.
▷ Saída: Avaliação de P em G, P(G).

1.  início
2.      n_max ← max_{r∈R} {|V(H_1)| : r = (H_1, H_2)}
3.      S ← {v ∈ V : d(v) ≤ d}
4.      enquanto S ≠ ∅ faça
5.          selecione v : v ∈ S
6.          r ← d-DESCOBRÍVEL(v, G, R)
7.          se r ≠ ∅ então                                          // Aplique r
8.              selecione G_1 : G_1 ⊆ G, v ∈ G_1, G_1 ≈ H_1, r = (H_1, H_2)
9.              selecione G_2 : G_2 ≈ H_2, r = (H_1, H_2)
10.             // Remoção de vértices internos e arestas de G_1
11.             V ← V \ {u ∈ V(G_1) : u ∈ Int(G_1)}
12.             E ← E \ E(G_1)
13.             S ← S \ {u ∈ V(G_1) : u ∈ Int(G_1)}
14.             // Adição de vértices internos e arestas de G_2
15.             V ← V ∪ {u ∈ V(G_2) : u ∈ Int(G_2)}
16.             E ← E ∪ E(G_2)
17.             S ← S ∪ {u ∈ V(G_1) : d(u) ≤ d
18.             para cada t ∈Term(G_2) faça
19.                 L ← LISTAADJ(t)
20.                 para cada (t, w) ∈ L : L mudou dentro da distância d faça
21.                     se d(w) ≤ d então S ← S ∪ {w}
22.         S ← S \ {v}
23.     se G ∈ I então retorna verdadeiro
```

Figura 4.8. Algoritmo para decisão de uma propriedade de grafos P dado um sistema de redução especial para P, Fase 1.

Figura 4.9. O primeiro passo é verificar a condição 2(a) da Definição 37 para G, na linha 25. Se a condição é satisfeita, então novas regras de redução são aplicadas a G. Ao final da segunda fase, G é realmente irredutível, já que todas as regras em $\mathcal{R}$ foram verificadas nas duas fases, e, como $\mathcal{R}$ é completo para G, pode-se decidir pela satisfação de P para G na linha 48.

Vale notar que o conjunto de regras de redução aplicáveis a G na segunda fase são tais que todos os lados esquerdos destas são grafos zero-terminais (linha 26), já que as outras regras já foram testadas na Fase 1. Para tanto, são construídas três estruturas de dados especiais, L_r (linhas 29 a 31), i_r (linha 33) e I_r (linha 34).

A lista L_r corresponde ao conjunto de grafos isomorfos às componentes do lado esquerdo de r, sem repetições. Posto de outra forma, L_r corresponde ao sistema de representantes distintos para as classes de equivalência definidas por isomorfismo nas

▷ **Algoritmo** REDUÇÃO$(G, \mathcal{R}, I)$ — Fase 2
▷ **Entrada:** Um grafo $G = (V, E)$ e um sistema de redução especial $(\mathcal{R}, I)$ para uma propriedade de grafos P e com parâmetros $n_{\min}$ e d.
▷ **Saída:** Avaliação de P em G, $P(G)$.

24. $S' \leftarrow$ COMPONENTES(G)
25. **se** $\exists C \in S' : |V(C)| \geq n_{\min}$ **então retorna** falso
26. $\mathcal{R}^0 \leftarrow \{r = (H_1, H_2) \in \mathcal{R} : \text{Term}(H_1) = \emptyset\}$
27. **para** cada $r = (H_1, H_2) \in \mathcal{R}^0$ **faça**
28. $\mathcal{C} \leftarrow$ COMPONENTES(H_1)
29. $L_r \leftarrow \emptyset$
30. **para** cada $C \in \mathcal{C}$ **faça**
31. **se** $\nexists C' \in L_r : C \approx C'$ **então** $L_r \leftarrow L_r \cup \{C\}$
32. **para** cada $H \in L_r$ **faça**
33. $i_r(H) \leftarrow |\{C \in \mathcal{C} : C \approx H\}|$
34. $I_r(H) \leftarrow \emptyset$
35. **enquanto** $S' \neq \emptyset$ **faça**
36. **selecione** $C : C \in S'$
37. **para** cada $r \in \mathcal{R}^0$ **faça**
38. **se** $\exists H \in L_r : H \approx C$ **então** $I_r(H) \leftarrow I_r(H) \cup \{C\}$
39. **se** $\exists r = (H_1, H_2) \in \mathcal{R}^0 : \forall H' \in L_r : |I_r(H')| \geq i_r(H')$ **então** `// Aplique r`
40. **para** cada $H' \in L_r$ **faça**
41. **selecione** $L(H') : L(H') \subseteq I_r(H'), |L(H')| = i_r(H')$
42. $G \leftarrow G \setminus_{\approx} L(H')$
43. **para** cada $r' \in \mathcal{R}^0$ **faça**
44. $I_{r'}(H') \leftarrow I_{r'}(H') \setminus_{\approx} L(H')$
45. $G \leftarrow G \cup H_2$
46. $S' \leftarrow S' \cup$ COMPONENTES(H_2)
47. $S' \leftarrow S' \setminus \{C\}$
48. **se** $G \in I$ **então retorna** verdadeiro **senão retorna** falso
49. **fim**

Figura 4.9. Algoritmo para decisão de uma propriedade de grafos P dado um sistema de redução especial para P, Fase 2.

componentes do lado esquerdo de r. Ainda considerando isomorfismo como uma relação de equivalência, pode-se definir uma extensão ao operador de diferença entre conjuntos para grafos, $\setminus_{\approx}$: denota-se por $G \setminus_{\approx} H$, G e H grafos desconexos, a retirada de G, para cada componente C de H, de uma componente C' isomorfa à C. O operador $\setminus_{\approx}$ é utilizado nas linhas 42 e 44. Vale notar que ambos os operandos podem conter componentes isomorfas, e que a operação pode ser aplicada também para conjuntos ou listas de grafos.

As listas i_r e I_r são relacionadas a cada elemento H em L_r: $i_r(H)$ armazena o número de componentes isomorfas a H no lado esquerdo de r, enquanto que $I_r(H)$ tem como objetivo guardar as componentes de G que são isomorfas a H, sendo inicialmente vazia.

A complexidade de REDUÇÃO, assim como a corretude, é definida pelo seguinte teorema, cuja prova encontra-se em (de Fluiter 1997):

Teorema 4.20 (de Fluiter [1997]). *Dados uma propriedade de grafos P e um sistema de redução especial $(\mathcal{R}, I)$ para P, o algoritmo* REDUÇÃO *reconhece corretamente grafos para os quais P vale e usa complexidade de tempo e espaço $O(|V(G)|)$, onde G é qualquer grafo dado como entrada.*

4.6 Algoritmos para problemas em grafos com largura em árvore limitada

Pode-se demonstrar que uma decomposição em árvore de um grafo $G = (V, E)$ pode ser facilmente convertida — ou seja, em tempo linear — em uma decomposição em árvore *agradável* $D = (\mathcal{X} = \{X_i\}_{i \in I}, T = (I, F))$ onde a árvore T é enraizada e binária e os nós de T são de quatro tipos:

folha: nós i do tipo folha são folhas em T e tais que $|X_i| = 1$.

introdução: nós i de introdução têm um filho j e satisfazem $X_i = X_j \cup \{v\}$, para algum vértice $v \in V$.

saída: nós i de saída têm um filho j e satisfazem $X_i = X_j \setminus \{v\}$, para algum vértice $v \in V$.

junção: nós de junção i têm dois filhos j_1 e j_2 e tais que $X_i = X_{j_1} = X_{j_2}$.

Como Bodlaender (1997) observa, a consideração de decomposições em árvore agradavéis geralmente não implica em novas possibilidades algorítmicas quando comparado com decomposições normais; no entanto, facilita enormemente o projeto de algoritmos, por se tratar de uma árvore binária. Além disso, pode-se esperar a obtenção de melhores fatores constantes escondidos nas complexidades dos algoritmos.

Em (Bodlaender 1997) é apresentada uma técnica para tratamento de problemas em grafos com largura em árvore limitada, baseada em *tabelas de caracterizações de soluções parciais*. Dada uma decomposição em árvore enraizada $D = (\mathcal{X}, T)$ de um grafo G, para cada nó i de T calcula-se uma *caracterização*, numa ordem de baixo para cima, ou seja, iniciando nas folhas e percorrendo os pais até atingir a raiz de T. Esta estratégia foi inicialmente apresentada em (Arnborg e Proskurowski 1989), e consiste de uma abordagem semelhante à programação dinâmica.

De um modo geral, um algoritmo baseado nesta técnica possui a seguinte estrutura, onde k é uma constante e limitante superior da largura em árvore do grafo de entrada $G = (V, E)$:

1. Encontre uma decomposição em árvore D de G de largura no máximo k.
2. Transforme D numa decomposição em árvore agradável $D' = (\mathcal{X} = \{X_i\}_{i \in I}, T = (I, F))$ de largura no máximo k e onde r é a raiz de T.

3. Calcule para cada nó $i \in I$ uma certa tabela. Para a obtenção de cada tabela deve-se usar somente as tabelas já calculadas para os filhos de i, o tipo de nó de i — folha, introdução, saída e junção — e a informação sobre G restrita a X_i.

4. A solução do problema encontra-se na tabela de r, a raiz de T.

Como Bodlaender (1997) observa, esta estratégia pode ainda ser usada para versões construtivas do problema, onde uma solução é construída na medida em que a árvore é percorrida de baixo para cima, juntamente com as tabelas, ou uma outra fase é necessária para construir a solução a partir das tabelas já prontas.

Para construção das tabelas, suponha que se queira resolver um certo problema $\mathcal{P}$, onde se pode assumir que $\mathcal{P}$ é uma propriedade de grafos. O projeto do algoritmo segue os seguintes passos:

1. Defina a noção de *solução*. Para alguns problemas esta pode ser bem clara, como no caso de MAX CONJUNTO INDEPENDENTE: um conjunto S de vértices não adjacentes entre si em um grafo G tal que S tem cardinalidade máxima dentre todos os outros conjuntos independentes em G.

2. Defina a noção de *solução parcial*. A idéia de solução parcial é intimamente relacionada a de grafo terminal, ou seja, assim como uma solução parcial deve contribuir para a formação de uma solução (geral), a operação de união entre grafos terminais deve construir o grafo de entrada G. Desta forma, uma solução parcial corresponde a uma solução particular do problema quando restrita a um subgrafo terminal de G. Como exemplo, uma solução parcial para MAX CONJUNTO INDEPENDENTE seria um conjunto independente máximo em H, um subgrafo terminal de G.

3. Defina a noção de *extensão de uma solução parcial*. A extensão deve descrever um modo de como obter uma solução a partir de soluções parciais. Para MAX CONJUNTO INDEPENDENTE, um conjunto S é uma solução para G e uma extensão de um conjunto S' para um subgrafo terminal H de G se e somente se S' é uma restrição de S em H.

4. Defina a noção de *característica de uma solução parcial*. Como Bodlaender descreve, equivale ao que se deve saber sobre uma solução parcial de modo a julgar a sua capacidade de extender para uma solução.

5. Um *conjunto completo de características* para um grafo terminal H é o conjunto de todas as características das soluções parciais em H. O conjunto completo de características de um grafo G_i para um nó i de uma decomposição em árvore agradável é também chamado *conjunto completo para* i.
   Mostre agora, para cada um dos quatro tipos de nós para i, o conjunto completo para i pode ser calculado eficientemente (em tempo polinomial ou constante), dados os conjuntos completos para todos os filhos de i e assumindo que existem no máximo $k + 1$ terminais em cada um dos grafos terminais envolvidos.

6. Mostre que o problema pode ser decidido eficientemente dado um conjunto completo para a raiz r da decomposição em árvore agradável.

Bodlaender (1997) fornece ainda definições mais formais, denotando por $sol_{\mathcal{P}}$, $solp_{\mathcal{P}}$, $ex_{\mathcal{P}}$ as relações para os conceitos de solução, solução parcial, e extensão, respectivamente e por $car_{\mathcal{P}}$ uma função correspondente ao conceito de característica. Desta forma, $sol_{\mathcal{P}}$ é uma relação que tem como argumentos um grafo G e uma solução s — que pode ser representada como um vetor ou uma cadeia de caracteres, por exemplo — de modo que, para todo grafo G, $\mathcal{P}(G)$ equivale a $\exists s : sol_{\mathcal{P}}(G, s)$. De modo análogo, $solp_{\mathcal{P}}$ admite dois argumentos, um grafo terminal H e uma solução parcial s'. A extensão $ex_{\mathcal{P}}$ vincula os dois conceitos anteriores, sendo uma relação com os quatro argumentos de $sol_{\mathcal{P}}$ e $solp_{\mathcal{P}}$: G um grafo, s uma solução, H um grafo terminal e s' uma solução parcial. Uma extensão deve satisfazer:

$$\forall G, s, H, s' : ex_{\mathcal{P}}(G, s, H, s') \implies \exists H' : G = H \oplus H' \wedge sol_{\mathcal{P}}(G, s) \wedge solp_{\mathcal{P}}(H, s')$$

e também

$$\forall G, s, H, s' : (sol_{\mathcal{P}}(G, s) \wedge (G = H \oplus H')) \implies \exists s' : solp_{\mathcal{P}}(H, s') \wedge ex_{\mathcal{P}}(G, s, H, s')$$

expressando que toda solução admite uma solução parcial em qualquer subgrafo terminal.

A função $car_{\mathcal{P}}$ tem como argumentos H, um subgrafo terminal, e s uma solução parcial, definidos quando $solp_{\mathcal{P}}(H, s)$ é verdadeiro. A característica $car_{\mathcal{P}}$ deve satisfazer, para todos os grafos terminais H, H' e H'', e todas as soluções parciais s e s':

$$car_{\mathcal{P}}(H, s) = car_{\mathcal{P}}(H', s') \implies$$
$$(\exists s'' : ex_{\mathcal{P}}(H \oplus H'', s'', H, s)) \Leftrightarrow (\exists s''' : ex_{\mathcal{P}}(H' \oplus H'', s''', H', s'))$$

O conjunto completo de características de um grafo terminal H é definido como $comp_{\mathcal{P}}(H) = \{car_{\mathcal{P}}(H, s) : solp_{\mathcal{P}}(H, s)\}$.

Como exemplo de aplicação mais notória, o algoritmo apresentado em (Bodlaender e Kloks 1996) é baseado nesta estrutura, com especificação de solução, solução parcial, extensão, característica e conjunto completo de características para, inicialmente, o problema de k-LARGURAEMCAMINHO, sendo os conceitos estendidos para o problema de k-LARGURAEMÁRVORE. Os conjuntos completos de características foram definidos para nós de uma decomposição em árvore agradável de um grafo de entrada G. Em (Bodlaender 1997) é apresentado um exemplo de aplicação para o problema de 3-partição do conjunto de vértices de um grafo G.

Assim como na técnica de redução a dificuldade maior é a obtenção das regras de redução e de um sistema de redução especial, aqui também a dificuldade está na especificação de conceitos adequados para a aplicação da técnica.

Capítulo 5

Conclusões e Recomendações

A proposta deste texto é realizar uma pesquisa do estado da arte em decomposições em árvore, e em algoritmos para sua obtenção. Além disso, o texto propõe-se a abordar tais aspectos em tom introdutório, de modo a facilitar a sua familiarização pelo leitor. De fato, pretende-se que o texto sirva como fonte de referência altamente recomendada, atuando como ponto de partida para vários pesquisadores. A partir do que foi discutido principalmente nos Capítulos 3 e 4, pode-se julgar que tais objetivos foram satisfatoriamente alcançados.

Embora todo o texto tenha favorecido uma abordagem simples e eficiente — de tal forma que o leitor possa assimilar mais rapidamente os conceitos e resultados — no Capítulo 3, em particular, procurou-se contribuir com provas mais refinadas, facilitadas, sendo portanto de grande valia para iniciantes pelo seu caráter didático. Optou-se por uma abordagem gradativa, partindo de casos específicos e culminando com resultados mais profundos, no sentido em que permitem ao pesquisador perceber, em âmbito mais geral, as características estruturais de grafos com largura em árvore limitada. Além de contribuir com essa visão mais abrangente, as últimas seções do Capítulo 3 retratam o estado da arte em termos de resultados teóricos para decomposição em árvore, não deixando de realçar os principais teoremas, suas relações e conseqüências.

No Capítulo 4 foram considerados os aspectos mais práticos, sendo apresentados algoritmos eficientes para obtenção de decomposições em árvore. Seguindo uma abordagem semelhante a do Capítulo 3, os algoritmos são discutidos gradativamente, partindo de casos particulares e prosseguindo para técnicas mais refinadas. Os algoritmos mais importantes atualmente — retratando também o estado da arte em termos de técnicas e resultados — foram divididos em dois paradigmas: o construtivo e o redutivo. No primeiro, técnicas tradicionais de algoritmos em grafos são empregadas com base em recursão, enquanto que no segundo é usada uma técnica de redução, baseada em lógica de segunda ordem e álgebra de grafos. Embora os algoritmos tenham sido assim divididos de modo a favorecer uma abordagem didática, eles não são de forma alguma independentes; de fato,

é até possível estabelecer uma relação de eqüivalência entre eles. Vale observar que tais algoritmos foram desenvolvidos com base em resultados obtidos para uma vasta gama de aplicações, sendo generalizados com base, principalmente, nestes dois paradigmas.

A maioria dos algoritmos conta com uma versão formalizada, apresentada numa pseudo-linguagem de programação. A idéia de tal representação visa facilitar uma visão algorítmica voltada para implementação. Assim como a visão algorítmica de decomposição em árvore veio enriquecer ainda mais a percepção dos conceitos e resultados teóricos — principalmente nos casos em que provas algorítmicas são fornecidas — um esforço de implementação dos algoritmos aqui discutidos tem grande efeito esclarecedor e didático. De fato, um produto secundário deste trabalho de pesquisa foi o desenvolvimento de um ambiente interativo para e com implementação dos principais algoritmos encontrados no texto, ou seja, além da implementação de rotinas e algoritmos, foram disponibilizadas ferramentas para construção de decomposições em árvore.

Uma das principais recomendações para trabalhos futuros contempla, então, o prosseguimento do trabalho de implementação de modo a acrescentar mais ferramentas, rotinas e algoritmos. Desta forma, este texto pretende ser também uma referência inicial para pesquisa de novos algoritmos. Outras recomendações podem sugerir ainda o desenvolvimento de novas referências para decomposição em árvore em língua portuguesa.

Referências Bibliográficas

ABRAHAMSON, K. R. E FELLOWS, M. R. (1993). "Finite automata, bounded treewidth and wellquasiordering". In: *Graph Structure Theory, Contemporary Mathematics*, Volume 147, pp. 539–564. Providence, EUA: Am. Math. Soc.

ALON, N., SEYMOUR, P., E THOMAS, R. (1990). "A separator theorem for graphs with an excluded minor and its applications". In: *Proceedings of the 22nd annual ACM Symposium on Theory of Computing*, pp. 293–299. ACM.

ARNBORG, S. (1985). "Efficient algorithms for combinatorial problems on graphs with bounded decomposability — a survey". *BIT* (25): 2–23.

ARNBORG, S., CORNEIL, D. G., E PROSKUROWSKI, A. (1987). "Complexity of finding embeddings in a k-tree". *SIAM J. Alg. Discrete Math.* (8): 277–284.

ARNBORG, S., COURCELLE, B., PROSKUROWSKI, A., E SEESE, D. (1993). "An algebraic theory of graph reduction". *J. Assoc. Comput. Mach. 40*(5): 1134–1164.

ARNBORG, S., LAGERGREN, J., E SEESE, D. (1991). "Easy problems for tree-decomposable graphs". *J. Algorithms* (12): 308–340.

ARNBORG, S. E PROSKUROWSKI, A. (1986). "Characterization and recognition of partial 3-trees". *SIAM J. Alg. Discrete Math.* (7): 305–314.

ARNBORG, S. E PROSKUROWSKI, A. (1989). "Linear time algorithms for np-hard problems restricted to partial k-trees". *Discrete Applied Math.* (23): 11–24.

ARNBORG, S. E PROSKUROWSKI, A. (1990). "Canonical representation of partial 2- and 3-trees". In: *Scandinavian Workshop on Algorithm Theory*, pp. 310–319.

BERGE, C. (1973). *Graph and Hypergraphs*. Amsterdam, Holanda: North Holland.

BODLAENDER, H. L. (1986). *Classes of Graphs with Bounded Treewidth*. Relatório técnico, Department of Computer Science, Utrecht University, Utrecht, Holanda.

BODLAENDER, H. L. (1988). "Dynamic programming algorithms on graphs with bounded tree-width". In: *Proceedings of the 15th International Colloquium on Automata, Languages and Programming*, Volume 317 de *Lecture Notes in Computer Science*, Berlin, Alemanha, pp. 105–119. Springer-Verlag.

BODLAENDER, H. L. (1993a). "Complexity of path-forming games". *Theoret. Comput. Sci.* (110): 215–245.

BODLAENDER, H. L. (1993b). "A linear time algorithm for finding tree-decompositions of small treewidth". In: *Proceedings of the 25th annual ACM Symposium on Theory of Computing*, San Diego, California, EUA, pp. 226–234. ACM Press.

BODLAENDER, H. L. (1993c). "A tourist guide through treewidth". *Acta Cybern.* (11): 1–23.

BODLAENDER, H. L. (1994). "Improved self-reduction algorithms for graphs with bounded treewidth". *Discrete Appl. Math.* (54): 101–115.

BODLAENDER, H. L. (1996). "A linear time algorithm for finding tree-decompositions of small treewidth". *SIAM Journal on Computing 25*: 1305–1317.

BODLAENDER, H. L. (1997). "Treewidth: Algorithmic techniques and results". In: I. Privara e P. Ruzicka (Eds.), *Proceedings 22nd International Symposium on Mathematical Foundations of Computer Science, MFCS'97, Lecture Notes in Computer Science*, Volume 1295, Berlin, Alemanha, pp. 29–36. Springer Verlag.

BODLAENDER, H. L. (2000). Necessary edges in k-chordalizations of graphs.

BODLAENDER, H. L., GILBERT, J. R., HAFSTEINSSON, H., E KLOKS, T. (1995a). "Approximating treewidth, pathwidth, and minimum elimination tree height". *J. Algorithms* (18): 238–255.

BODLAENDER, H. L. E HAGERUP, T. (1995). "Parallel algorithms with optimal speedup for bounded treewidth". In: *Proceedings of the 22nd International Colloquium on Automata, Languages and Programming*, Volume 944, Berlin, Alemanha, pp. 268–279. Springer Verlag.

BODLAENDER, H. L. E KLOKS, T. (1996). "Efficient and constructive algorithms for the pathwidth and treewidth of graphs". *J. Algorithms* (21): 358–402.

BODLAENDER, H. L., KLOKS, T., E KRATSCH, D. (1995b). "Treewidth and pathwidth of permutation graphs". *SIAM J. Discrete Math.* (8): 606–616.

BODLAENDER, H. L. E MÖHRING, R. H. (1993). "The pathwidth and treewidth of cographs". *SIAM J. Discrete Math.* (6): 181–188.

BODLAENDER, H. L. E THILIKOS, D. M. (1999). "Graphs with branchwidth at most three". *J. Algorithms* (32): 167–194.

BONDY, J. A. E MURTY, U. S. R. (1976). *Graph Theory with Applications*. Londres, Inglaterra: McMillan.

BORIE, R. B., PARKER, R. G., E TOVEY, C. A. (1992). "Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families". *Algorithmica* (7): 555–582.

COURCELLE, B. (1990). "The Monadic Second-order Logic of graphs I: Recognizable sets of finite graphs". *Information and Computation 85*.

DE FLUITER, B. (1997). *Algorithms for Graphs of Small Treewidth*. Ph. D. thesis, Department of Computer Science, Universiteit Utrecht, Utrecht, Holanda.

DE FLUITER, B. E BODLAENDER, H. L. (1995). *Reduction Algorithms for Graphs with Small Treewidth*. Relatório Técnico UU-CS-1995-37, Department of Computer Science, Universiteit Utrecht, Utrecht, Holanda.

DE FLUITER, B. E BODLAENDER, H. L. (1996). "Reduction algorithms for graphs with small treewidth". In: *Proceedings of the 2nd Annual International Conference on Computing and Combinatorics, COCOON'96*, Volume 1090, pp. 199–208. Springer Verlag, Lecture Notes in Computer Science.

DIESTEL, R. (2000). *Graph Theory*. New York, NY, EUA: Springer Verlag.

DIMITRIOU, T. E IMPAGLIAZZO, R. (1996). "Towards an analysis of local optimization algorithms". In: *Proceedings of the 28th annual ACM Symposium on Theory of Computing*, Philadelphia, Pennsylvania, EUA, pp. 304–313. ACM Press.

DJIDJEV, H. E REIF, J. (1991). "An efficient algorithm for the genus problem with explicit construction of forbidden subgraphs". In: *Proceedings of the 23rd annual ACM Symposium on Theory of Computing*, New Orleans, Louisiana, EUA, pp. 337–347. ACM Press.

ELLIS, J. A., SUDBOROUGH, I. H., E TURNER, J. (1994). "The vertex separation and search number of a graph". *Inform. and Comput.* (113): 50–79.

EVANS, J. R. E MINIEKA, E. (1992). *Optimization Algorithms for Networks and Graphs*. E.U.A.: Marcel Dekker Inc.

FELLOWS, M. R. E LANGSTON, M. A. (1989). "On search decision and the efficiency of polynomial-time algorithms (extended abstract)". In: *Proceedings of the 21st annual ACM Symposium on Theory of Computing*, Seattle, Washington, EUA, pp. 501–512. ACM Press.

FELLOWS, M. R. E LANGSTON, M. A. (1994). "On search, decision and the efficiency of polynomial-time algorithms". *J. Comput. System. Sci.* (49): 769–779.

FREDERICKSON, G. N. (1991). "Planar graph decomposition and all pairs shortest paths". *Journal of the ACM 38*(1): 162–204.

GAVOILLE, C., PELEG, D., PÉRENNES, S., E RAZ, R. (2001). "Distance labeling in graphs". In: *Proceedings of the 12th annual ACM Symposium on Discrete Algorithms*, Washington, D.C., EUA, pp. 210–219. ACM Press.

GOLUMBIC, M. C. (1980). *Algorithmic Graph Theory and Perfect Graphs*. Nova York, E.U.A.: Academic Press.

HABIB, M. E MÖHRING, R. H. (1994). "Treewidth of cocomparibility graphs and a new order-theoretic parameter". *ORDER* (1): 47–60.

KLOKS, T. (1993). *Treewidth of Circle Graphs*. Relatório técnico, Department of Computer Science, Utrecht University, Utrecht, Holanda.

KLOKS, T. (1994). "Treewidth — computations and approximations". In: *Lecture Notes in Computer Science*, Volume 842. Springer-Verlag.

KLOKS, T. E KRATSCH, D. (1994). "Finding all minimal separators of a graph". In: *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science*, Volume 775 de *Lecture Notes in Computer Science*, Berlin / Nova York, pp. 759–768. Springer-Verlag.

KLOKS, T. E KRATSCH, D. (1995). "Treewidth of chordal bipartite graphs". *J. Algorithms* (19): 266–281.

KOBLER, D. E ROTICS, U. (2001). "Polynomial algorithms for partitioning problems on graphs with fixed clique-width (extended abstract)". In: *Proceedings of the 12th annual ACM Symposium on Discrete Algorithms*, Washington, D.C., EUA, pp. 468–490. ACM Press.

KOSTER, A. M., BODLAENDER, H. L., E VAN HOESEL, S. P. (2001). "Treewidth: Computational experiments". *Electronic Notes in Discrete Mathematics 8*.

LAGERGREN, J. (1996). "Efficient parallel algorithms for graphs of bounded tree-width". *J. Algorithms* (20): 20–44.

LAGERGREN, J. E ARNBORG, S. (1991). "Finding minimal forbidden minors using a finite congruence". In: *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, Volume 510 de *Lecture Notes in Computer Science*, Berlin / Nova York, pp. 533–543. Springer-Verlag.

MAHESHWARI, A. E ZEH, N. (2001). "I/o-efficient algorithms for graphs of bounded treewidth". In: *Proceedings of the 12th annual ACM Symposium on Discrete Algorithms*, Washington, D.C., EUA, pp. 89–90. ACM Press.

MATOUSEK, J. E THOMAS, R. (1991). "Algorithms finding tree-decompositions of graphs". *J. Algorithms* (12): 1–22.

MATOUSEK, J. E THOMAS, R. (1992). "On the complexity of finding iso- and other morphisms for partial k-trees". *Discrete Mathematics* (108): 343–364.

PAPADIMITRIOU, C. H. E STEIGLITZ, K. (1983). *Combinatorial Optimization: Algorithms and Complexity*. New Jersey, E.U.A.: Prentice Hall, Englewood Cliffs.

PERKOVIC, L. E REED, B. (2000). "An improved algorithm for finding tree decompositions of small width". *International Journal of Foundations of Computer Science 11*(3): 365–371.

REED, B. A. (1992). "Finding approximate separators and computing tree width quickly". In: *Proceedings of the 24th annual ACM Symposium on Theory of Computing*, Victoria, British Columbia, Canada, pp. 221–228. ACM Press.

REED, B. A. (1997). *Surveys in Combinatorics*, Volume 315, Capítulo Tree Width and Tangles: A New Conectivity Measure and Some Applications, R. Bailey (ed.), pp. 87–162. Cambridge University Press.

ROBERTSON, N. E SEYMOUR, P. D. (1983). "Graph minors. I. Excluding a forest". *J. Combin. Theory Ser. B.* (35): 39–61.

ROBERTSON, N. E SEYMOUR, P. D. (1985). *Surveys in Combinatorics*, Capítulo Graph Minors — A Survey, pp. 153–171. Cambridge, RU: Cambridge Univ. Press.

ROBERTSON, N. E SEYMOUR, P. D. (1986). "Graph minors. II. Algorithmic aspects of tree-width". *J. Algorithms* (7): 309–322.

ROBERTSON, N. E SEYMOUR, P. D. (1991). "Graph minors. X. Obstructions to tree-decomposition". *J. Combin. Theory Ser. B.* (52): 153–190.

ROBERTSON, N. E SEYMOUR, P. D. (1995). "Graph minors. XIII. The disjoint paths problem". *J. Combin. Theory Ser. B.* (63): 65–110.

SANDERS, D. P. (1996). "On linear recognition of tree-width at most four". *SIAM J. Discrete Math.* (9): 101–117.

WILSON, R. J. (1985). *Introduction to Graph Theory*. Nova York, E.U.A.: Longman Scientific & Technical.