COPPE/UFRJ

BANDWIDTH EM GRAFOS

Vitor Augusto Ferreira Santa Rita

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadora: Márcia Rosana Cerioli

Rio de Janeiro
Setembro de 2010

BANDWIDTH EM GRAFOS

Vitor Augusto Ferreira Santa Rita

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Profª. Márcia Rosana Cerioli, D.Sc.

Profª. Claudia Marcela Justel, D.Sc.

Prof. Nelson Maculan Filho, D.Habil.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2010

Santa Rita, Vitor Augusto Ferreira

BANDWIDTH em grafos/Vitor Augusto Ferreira Santa Rita. – Rio de Janeiro: UFRJ/COPPE, 2010.

XII, 88 p.: il.; 29, 7cm.

Orientadora: Márcia Rosana Cerioli

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 85 – 88.

1. Bandwidth. 2. Algoritmos. 3. Grafos. I. Cerioli, Márcia Rosana. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha esposa, Íris.

Agradecimentos

Agradeço aos meus pais, Paulo e Valéria, pela dedicação incondicional na criação e na educação ao longo da minha vida. E ao meu irmão, Isaac, pelo seu companheirismo e amizade.

Agradeço a minha esposa, Íris, pelo apoio, pela paciência, pelo incentivo e pela compreensão em todos os momentos em que precisei me privar de estar ao seu lado, em virtude dos meus estudos tanto no mestrado quanto na minha graduação, quando não tive oportunidade de fazer este agradecimento.

Gostaria de agradecer, em especial, à minha orientadora, prof[a]. Márcia Cerioli, por tudo que eu pude aprender com ela nesses anos.

Agradeço aos professores Cláudia Justel e Nelson Maculan por aceitarem fazer parte da minha banca examinadora.

Gostaria de agradecer ao meu amigo Daniel Posner pela amizade e pela sua ajuda ao longo do mestrado nas discussões e revisões do texto da dissertação.

Gostaria de agradecer também ao Programa de Engenharia de Sistemas e Computação da Coordenadoria de Programas de Pós-graduação em Engenharia da Universidade Federal do Rio de Janeiro pela oportunidade de estudar num centro de excelência.

Gostaria de agradecer ao Centro Tecnológico do Exército pelo disponibilidade de tempo e pelo apoio para que eu pudesse realizar este mestrado.

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

BANDWIDTH EM GRAFOS

Vitor Augusto Ferreira Santa Rita

Setembro/2010

Orientadora: Márcia Rosana Cerioli

Programa: Engenharia de Sistemas e Computação

BANDWIDTH é um problema de otimização combinatória que busca minimizar a maior diferença dos rótulos dos vértices adjacentes de um grafo $G = (V, E)$, quando rotula-se os vértices de G com números naturais diferentes. Esse problema foi mostrado ser NP-completo, em 1976, e são conhecidas apenas algumas classes de grafos para as quais existe um algoritmo polinomial. Esta dissertação apresenta duas demonstrações de NP-completude para o problema, além de apresentar os principais algoritmo polinomiais existentes bem como dois algoritmo exponenciais exatos para a classe geral de grafos.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

BANDWIDTH IN GRAPHS

Vitor Augusto Ferreira Santa Rita

September/2010

Advisor: Márcia Rosana Cerioli

Department: Systems Engineering and Computer Science

BANDWIDTH is a combinatorial optimization problem that consists in minimizing the greatest difference among labels of adjacent vertices in a graph $G = (V, E)$, when those vertices are labeled with distinct natural numbers. This problem was shown to be NP-complete, in 1976, and only a few subclasses are known to have a polinomial time algorithm. This dissertation shows two NP-complete demonstrations of the problem, besides presenting the principals polinomial time algorithms of classes of graphs together with two exact exponential time algorithms to the general class of graphs.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O problema de *bandwidth*, ou problema de *largura de banda*, em grafos consiste em atribuir números naturais distintos aos vértices de um grafo de modo que a maior diferença entre os números atribuídos a vértices adjacentes seja mínima.

Este é um problema de otimização clássico que surgiu a partir da computação com matrizes simétricas esparsas, onde as operações são executadas de maneira mais eficiente quando aplicadas a uma permutação das linhas e das colunas da matriz, de modo que todas as entradas não nulas estejam próximas da diagonal principal. Define-se o *bandwidth* de uma matriz simétrica esparsa M como o maior natural k para o qual há um elemento não nulo em $M_{i,i+k}$, $i \in \mathbb{N}$.

O problema de *bandwidth* em matrizes é, na realidade, a versão na qual a noção de *bandwidth* começou a ser estudada, com vários artigos publicados na década de 60 (ver referências do artigo de Gibbs *et al* [9]). O problema de reduzir o *bandwidth* de uma matriz M consiste em encontrar uma matriz de permutação P tal que $M' = P \times M \times P^T$ tenha o menor *bandwidth*. Ao considerar que M é uma matriz simétrica $n \times n$ que representa um grafo $G = (V, E)$ — cada elemento $M_{ij} \neq 0$ se, e somente se, $v_i v_j \in E$ — e que a permutação de linhas e colunas representa os rótulos dados aos vértices de G, então o *bandwidth* da matriz é igual ao $bw(G)$. A versão matricial deste problema, contudo, não será abordada neste trabalho, mas apenas a formulação em grafos.

Dentre as várias aplicações deste problema, destacam-se: solução de sistemas lineares — tarefa necessária para resolver problemas de programação linear; solução de desenho de circuitos VLSI; solução de problemas de interconexão de redes; solução de uma classe de problemas conhecidos como "problemas de satisfação de restrições".

Uma explicação detalhada sobre cada um deles pode ser encontrada em Lai e Williams [25].

O problema de *bandwidth* em grafos foi provado ser NP-completo por Papadimitriou em 1976 [29] e permanece intratável mesmo para classes bem restritas de grafos, tais como árvores e split. Desta forma, muitos dos resultados conhecidos sobre o problema são relacionados a apresentação de heurísticas ou algoritmos aproximados. Nesta dissertação, optou-se por apresentar os principais resultados conhecidos sobre algoritmos exatos para o problema de *bandwidth*.

Esta dissertação está organizada como segue. Na próxima seção serão apresentadas as definições de grafos que são usadas ao longo dos próximos capítulos. No capítulo 2, é problema BANDWIDTH é formalmente descrito e os resultados exatos para grafos simples como ciclo, caminho, completo, bipartido completo são apresentados. Ainda no capítulo 2, são apresentados os limites inferiores e os resultados do problema PATHWIDTH associados a BANDWIDTH. No capítulo 3, são feitas duas demonstrações de NP-completude. No capítulo 4, são apresentados os principais resultados de algoritmos polinomiais para classes restritas de grafos. No capítulo 5, descrevem-se dois algoritmos exatos exponenciais para a classe geral: um $O(n^{k+1})$ e o outro $O(5^n)$, onde n é o número de vértices de entrada do algoritmo e k é o valor de *bandwidth* para o qual se deseja saber se o grafo admite uma rotulação menor ou igual a esse valor. E por fim, no capítulo 6, tem-se a conclusão do trabalho.

1.1 Definições básicas

Um **grafo** $G = (V, E)$ consiste de um conjunto finito V de **vértices** e um conjunto finito E de arestas. Cada **aresta** $e \in E$ é um par não ordenado $e = \{u, v\}$ com $u, v \in V$; os vértices u e v são chamados **extremos** da aresta e. Por simplicidade, o par não ordenado $\{u, v\}$ pode ser escrito simplesmente uv. O tamanho do conjunto de vértices e de arestas é, em geral, denotado por $|V| = n$ e $|E| = m$. Se $|V| = 1$ e $E = \emptyset$, o grafo é **trivial**.

Dois vértices u e v são **adjacentes** se $uv \in E$. O conjunto $Adj(u)$ contém os vértices adjacentes ao vértice u, isto é, $Adj(u) = \{v \in V \ : uv \in E\}$. O **grau** de u é $d(u) = |Adj(u)|$. Adicionalmente, a **vizinhança** de u é $N(u) = \{u\} \cup Adj(u)$. Um vértice v é **universal** se $N(u) = V$.

Um grafo H é **subgrafo** de G — e G é **supergrafo** de H —, representado por

$H \subseteq G$, se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Se $H \subsetneq G$, então H é **subgrafo próprio** de G. Para um subconjunto $A \subseteq V(G)$ de vértices, o **subgrafo induzido** por A é o grafo $G[A] = (A, E_A)$, onde $E_A = \{uv \in E(G) \ : u \in A \text{ e } v \in A\}$.

Seja $G = (V, E)$ um grafo. Uma sequência $[v_0, v_1, \ldots, v_k]$ de vértices de G é um **caminho** de v_0 até v_k em G se $v_{i-1}v_i \in E$ para $i = 1, 2, \ldots, k$. Uma sequência de vértices $[v_0, v_1, \ldots, v_k = v_0]$, $k \geq 2$, é um **ciclo** se $[v_0, v_1, \ldots, v_{k-1}]$ é um caminho cujos vértices são distintos e $v_{k-1}v_k \in E$. Um grafo é **acíclico** se não possui ciclo. O **comprimento**, ou **tamanho**, de um caminho ou ciclo $[v_0, v_1, \ldots, v_k]$ é k. Se não há aresta v_iv_j tal que i e j difiram por mais de uma unidade num ciclo $[v_0, v_1, \ldots, v_k = v_0]$, diz-se que o ciclo é **sem corda** ou **induzido**.

A **distância** entre dois vértices u e v em um grafo G é o tamanho do menor caminho de u até v. Enquanto que o **diâmetro** de G é a maior distância entre quaisquer dois vértices de G.

Um grafo G é **conexo** se entre quaisquer dois vértices u e v existe um caminho de u até v. Caso G não seja conexo, *i.e.* **desconexo**, então cada subgrafo conexo, induzido por um subconjunto maximal (em relação a inclusão de conjuntos) de vértices de G, é um **componente conexo**.

Uma **árvore** é um grafo conexo e acíclico.

Um grafo $G = (V, E)$ é **completo** se existe um aresta $uv \in E(G)$ para todo par de vértices u e v de $V(G)$. Um subconjunto $A \subseteq V(G)$ é uma **clique** se o subgrafo induzido por A é completo; e é um **conjunto independente** se nenhum par de vértices de A é adjacente em G.

Uma **tripla asteroidal** em um grafo $G = (V, E)$ é um conjunto independente de três vértices, tal que entre quaisquer dois deles existe um caminho P em G tal que nenhum vértice de P é adjacente ao terceiro vértice da tripla.

Um grafo $G = (V, E)$ é **bipartido** se o conjunto de vértices V pode ser particionado em dois conjuntos independentes de vértices, X e Y, tais que $V = X \cup Y$ e $X \cap Y = \emptyset$, onde toda aresta tem um extremo em X e o outro em Y. Grafos bipartidos são representados por $G = (X, Y, E)$, de forma a enfatizar a partição do conjunto de vértices. Se $G = (X, Y, E)$ for bipartido e existir $uv \in E$ para todos $u \in X$ e $v \in Y$, então G é **bipartido completo**.

O **complemento** de um grafo $G = (V, E)$ é o grafo $\overline{G} = (V, \overline{E})$, onde $\overline{E} = \{uv \in V \times V : u \neq v \text{ e } uv \notin E\}$.

Suponha que G e H são grafos com conjuntos de vértices disjuntos, *i.e.* $V(G) \cap$

$V(H) = \emptyset$. A **união** de G e H é o grafo $G \cup H$ onde

$$V(G \cup H) = V(G) \cup V(H) \text{ e } E(G \cup H) = E(G) \cup E(H).$$

E a **junção** de G e H é o grafo $G + H = J$ onde

$$V(J) = V(G) \cup V(H) \text{ e } E(J) = E(G) \cup E(H) \cup \{uv \ : u \in V(G) \text{ e } v \in V(H)\}.$$

É fácil ver que existe uma relação entre a união, o complemento e a junção de grafos. De fato:

$$\overline{G + H} = \overline{G} \cup \overline{H} \tag{1.1}$$

Para representar um grafo computacionalmente existem duas opções usuais: matriz de adjacências e lista de adjacências.

Na matriz de adjacências, um grafo $G = (\{v_1, \ldots, v_n\}, E)$ é representado por duas estruturas de dados: um vetor para os vértices e uma matriz M para as arestas. O vetor tem tamanho igual a quantidade de vértices do grafo, com cada elemento do vetor representando um vértice; e a matriz M tem tamanho $n \times n$, na qual os elementos $M_{i,j}$ e $M_{j,i}$ são diferentes de zero se, e somente se, os vértice v_i e v_j são adjacentes.

Na lista de adjacências, o grafo $G = (\{v_1, \ldots, v_n\}, E)$ é representado por um vetor de vértices e várias listas para as arestas. O vetor é o mesmo utilizado na representação por matriz de adjacências; e, para cada vértice v_i, existe uma lista associada, na qual estão contidos todos os vértices v_j tal que $v_i v_j \in E$.

1.1.1 Notações

Dado um grafo G, seguimos a nomenclatura usual da teoria dos grafos e adotamos as seguintes notações para representar os parâmetros usuais de G:

- $\omega(G)$: tamanho da maior clique de G;
- $\alpha(G)$: tamanho do maior conjunto independente de G;
- $d(v)$: grau de um vértice v;
- $\delta(G)$: menor grau de um vértice de G;

- Δ: maior grau de um vértice de G;
- $dist_G(u, v)$: distância entre os vértices u e v de G;
- $diam(G)$: diâmetro de G;

Outros símbolos usados, inclusive para definir classes restritas de grafos, podem ser encontrados no livro de Golumbic [11].

Capítulo 2

Bandwidth em Grafos

Este capítulo apresenta a descrição formal do problema BANDWIDTH, bem como os resultados de valores exatos para grafos simples como ciclo, caminho, completo e bipartido completo, e limites inferiores que são usados nas demonstrações ao longo da dissertação. E uma seção é dedicada ao estudo do problema PATHWIDTH, para apresentar resultados também usados nas demonstrações dos capítulos seguintes.

2.1 Descrição formal do problema

Seja $G = (V, E)$ um grafo com $|V| = n$. Uma **rotulação** de G é uma função bijetora $f : V \to \{1, \ldots, n\}$. A **largura** de f em G é dada por

$$bw_f(G) = \max\{|f(u) - f(v)| : uv \in E\}.$$

enquanto que o **bandwidth** de G é

$$bw(G) = \min\{bw_f(G) : f \text{ é rotulação de } G\}.$$

Uma rotulação f de G é **ótima** se $bw_f(G) = bw(G)$. Se o domínio de f for um subconjunto de V, então a rotulação é **parcial**.

A figura 2.1 apresenta uma rotulação f de um grafo G. Neste caso, $bw_f(G) = 4$ por causa da aresta com rótulos 2 e 6 em seus extremos.

Já a figura 2.2 apresenta uma outra rotulação de G, de onde se conclui que $bw(G) \leq 2$. Como qualquer rotulação parcial de G deve, no melhor caso, atribuir

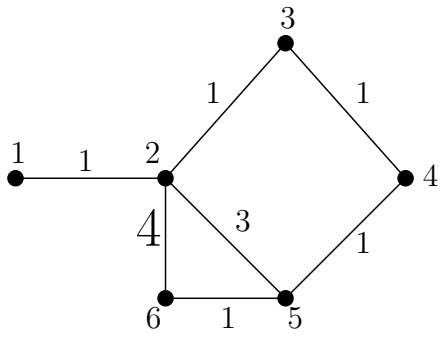

Figura 2.1: Um grafo G com uma rotulação f onde $bw_f(G) = 4$.

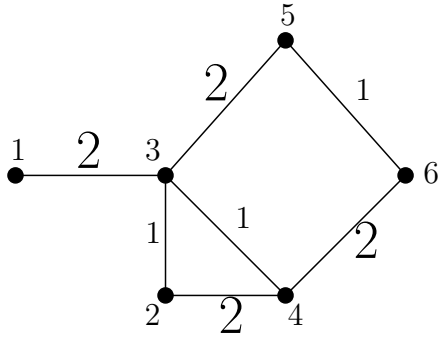

Figura 2.2: Um grafo G com uma rotulação ótima: $bw(G) = 2$.

rótulos consecutivos aos vértices da clique de tamanho 3, temos mais precisamente que $bw(G) = 2$. Assim, a rotulação apresentada na figura 2.2 é ótima.

Os grafos tratados nesta dissertação serão sempre conexos. Para um grafo desconexo, o valor de *bandwidth* é obtido pelo máximo do *bandwidth* de cada componente conexo.

A versão decisão do problema de *bandwidth* em grafos é descrita como segue:

Problema: BANDWIDTH
Instância: Um grafo G e um número inteiro positivo k.
Questão: $bw(G) \leq k$?

O problema BANDWIDTH foi mostrado ser NP-completo por Papadimitriou em 1976 [29], e permanece NP-completo até mesmo quando restrito a *caterpillar* com cabelos de tamanho 3 [28] ou a árvores com grau máximo 3 [6]. Além disso, existem poucas classes de grafos para as quais BANDWIDTH pode ser resolvido de maneira eficiente.

Classe	Complexidade	Referência
Bipartidos de permutação	$O(n^4 \log n)$	Heggernes *et al* 2008 [13]
Arco-circular	NP-completo	Kratsch e Stewart 2002 [24]
Split	NP-completo	Kloks *et al* 2000 [20]
Cobipartido	NP-completo	Kloks *et al* 1999 [22]
Cocomparabilidade	NP-completo	Kloks *et al* 1999 [22]
Grade	NP-completo	Díaz *et al* 1999 [5]
Chain	$O(n^2 \log n)$	Kloks *et al* 1998 [21]
P_4-esparso	Linear	Yan 1998 [36]
Tolerância	NP-completo	Hung *et al* 1998 [14]
Cografos	Linear	Yan 1997 [35]
Quasi-threshold	Linear	Chen *et al* 1996 [37]
Intervalo	$O(n \log n)$	Sprague 1994 [33]
	$O(nk)$	Kleitman e Vohra 1990 [19]
Theta	Polinomial	Peck e Shastri 1992 [30]
Caterpillar com cabelos ≤ 2	$O(n \log n)$	Assmann *et al* 1981 [1]
Árvores com $\Delta \leq 3$	NP-completo	Garey *et al* 1978 [6]
Classe geral de grafos	NP-completo	Papadimitriou 1976 [29]

Tabela 2.1: Classes de grafos e a complexidade do problema BANDWIDTH.

A tabela 2.1 exibe a complexidade de BANDWIDTH para uma lista de classes de grafos, organizada em ordem cronológica decrescente das datas de publicação dos

resultados. Já a tabela 2.2 apresenta classes de grafos para as quais BANDWIDTH é NP-completo, em virtude de ser NP-completo para alguma subclasse. Esta lista de classes foi obtida a partir dos livros de Golumbic [11] e Spinrad [32].

Classe	Subclasse
AT-free	Cobipartido
Bipartido	Árvore
Bipartido cordal	Árvore
Caminho	*Split*
Círculo	Árvore
Comparabilidade	Árvore
Cordal	*Split*
Disco unitário	Grid
Distância hereditária	Árvore
Doubly chordal	Árvore
Domination	*Split*
Dually chordal	Árvore
Fracamente cordal	Árvore
Fortemente cordal	Árvore
Interseção de discos	Grid
Número de intervalo	Arco-circular
Perfeitamente ordenável	*Split*
Perfeito	Árvore
Treewidth-k	Árvore

Tabela 2.2: Classes de grafos para as quais BANDWIDTH é NP-completo em virtude de existir uma subclasse para a qual BANDWIDTH também é um problema NP-completo.

Embora BANDWIDTH seja NP-completo, sua versão para k fixo é polinomial, como será visto na seção 5.1.

Problema: k-BANDWIDTH
Instância: Um grafo G.
Questão: $bw(G) \leq k$?

2.2 Valores exatos

Nesta seção, de forma a exemplificar raciocínios básicos e úteis na solução de BANDWIDTH, determinamos valores exatos para o *bandwidth* de grafos em classes restritas.

Para caminhos com n vértices, P_n, basta rotular os vértices com uma numeração sequencial ao longo do caminho. É fácil perceber que esta rotulação é ótima, pois resulta na equação (2.1) e este é o valor mínimo para a largura de uma rotulação.

$$bw(P_n) = 1 \tag{2.1}$$

Observamos que os grafos P_n são os únicos gráfos conexos G tais que $bw(G) = 1$. De fato, em qualquer rotulação f, vértices com grau maior do que 2 implicam em $bw_f(G) \geq 2$. Além disso, um vértice u tal que $f(u) = 1$ ou $f(u) = n$ também implica que $bw(G) \geq 2$.

Para grafos completos com n vértices, K_n, temos:

$$bw(K_n) = n - 1 \tag{2.2}$$

A equação (2.2) é apenas parte do resultado mais geral dado no lema 2.1.

Lema 2.1. *Um grafo G é completo, e somente se, $bw(G) = n - 1$.*

Demonstração. Se G é completo, $bw(G) = n - 1$ porque os rótulos 1 e n vão ser atribuídos a vértices adjacentes. Reciprocamente, se $bw(G) = n - 1$, então não existem vértices não adjacentes, pois caso contrário seria possível atribuir a eles os rótulos 1 e n, fazendo com que $bw(G) < n - 1$. □

Já para os ciclos com n vértices, C_n, temos:

$$bw(C_n) = 2 \tag{2.3}$$

Uma justificativa para a equação (2.3) pode ser obtida ao examinar a figura 2.3.

De forma geral, basta atribuir rótulo 1 a um vértice inicial qualquer, e atribuir sequencialmente os rótulos pares aos vértices de uma das metades do ciclo definida

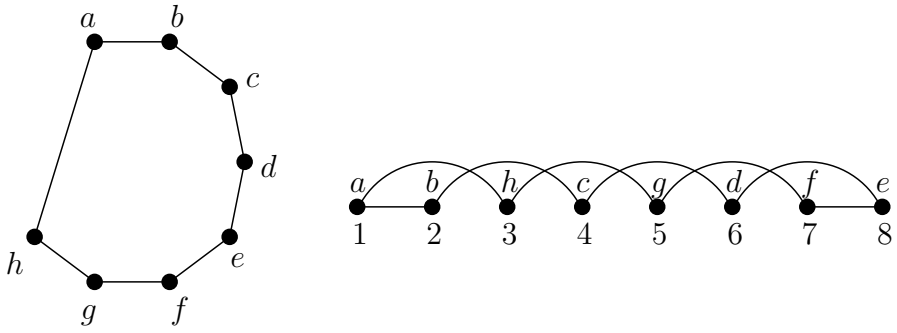

Figura 2.3: Rotulação do grafo C_8 mostrando que $bw(C_8) = 2$.

a partir do vértice inicial e os rótulos ímpares aos vértices da outra metade. É fácil perceber que esta rotulação é ótima, pois resulta em $bw(C_n) \leq 2$ e este é o valor mínimo para a largura de uma rotulação de um grafo que não é um caminho P_n.

Para grafos bipartidos completos, $K_{p,q}$ com $p \geq q$, temos:

$$bw(K_{p,q}) = \left\lceil \frac{p}{2} \right\rceil + q - 1 \tag{2.4}$$

Para provar o resultado anterior, seja a rotulação dos vértices de $K_{p,q} = (P, Q, E)$ como segue. Os vértices de P recebem os rótulos

$$\left\{1, 2, \ldots, \left\lceil \frac{p}{2} \right\rceil, \left\lceil \frac{p}{2} \right\rceil + q + 1, \ldots, n\right\}$$

e os vértices de Q recebem os rótulos

$$\left\{\left\lceil \frac{p}{2} \right\rceil + 1, \ldots, \left\lceil \frac{p}{2} \right\rceil + q\right\}.$$

Como $K_{p,q}$ possui todas as arestas uv com $u \in P$ e $v \in Q$, o valor de $bw(K_{p,q})$ da rotulação acima será o da equação (2.4). Basta mostrar que esse valor é ótimo. De fato, considere que o rótulo 1 seja dado a um vértice de P. Obviamente, o rótulo n será dado também a um vértice de P, para não haver aresta com extremos rotulados em 1 e n. Sejam x e y, respectivamente, o menor e o maior rótulos de Q. Suponha, agora, por contradição que exista uma rotulação de $K_{p,q}$ cujo valor de *bandwidth*

seja menor que o da equação (2.4). Isso implica que

$$\begin{aligned} y - 1 &< \left\lceil \frac{p}{2} \right\rceil + q - 1 \\ (p+q) - x &< \left\lceil \frac{p}{2} \right\rceil + q - 1 \end{aligned}$$

que resulta em

$$\begin{aligned} y - x &< 2(\left\lceil \frac{p}{2} \right\rceil + q - 1) + 1 - (p+q) \\ q \leq y - x + 1 &< 2(\left\lceil \frac{p}{2} \right\rceil + q - 1) + 2 - (p+q) \\ 0 &< 0, \end{aligned}$$

que é um absurdo.

2.3 Limites inferiores

A seguir, alguns limites inferiores para o valor de *bandwidth* em função de outros parâmetros do grafo são apresentados. Estes resultados são importantes nas provas de que os algoritmos apresentados para BANDWIDTH estão corretos.

A inequação (2.5) estabelece um limite inferior importante para o valor de *bandwidth* em grafos. Em particular, ele é usado na demonstração de alguns resultados da seção 5.1.

$$bw(G) \geq \left\lceil \frac{\Delta(G)}{2} \right\rceil \tag{2.5}$$

De fato, um vértice v com grau $d(v) = \Delta$ mais os adjacentes a ele induzem um grafo $G[\{v, Adj(v)\}]$ com $\Delta+1$ vértices, no qual serão usados $\Delta+1$ números naturais para rotulá-los. Sabe-se que v é universal nesse subgrafo. Se por contradição todos os vértices adjacentes a v forem receberem rótulos menor que $\frac{\Delta(G)}{2}$ unidades distantes do rótulo de v, não haveria rótulos suficientes para rotular todo o subgrafo. Então, é necessário que para ao menos um vértice adjacente a v seja dado um rótulo diferente do rótulo de v de ao menos $\frac{\Delta(G)}{2}$ unidades.

A inequação (2.6) exibe um resultado imediato sobre o problema: se um subgrafo H de um grafo G satisfaz $bw(H) \geq k$, então $bw(G) \geq k$. De fato, se $bw(G) < k$, então considerando uma rotulação ótima f de G, a rotulação parcial g restringindo

f aos vértices de H provaria que $bw(H) < k$, uma contradição

$$bw(G) \geq bw(H), \quad H \subseteq G \tag{2.6}$$

O resultado da equação (2.7) é uma aplicação da equação (2.6) a subgrafos induzidos.

$$bw(G) = \max\{bw(H) : H \text{ é subgrafo induzido de } G\} \tag{2.7}$$

As equações (2.8) e (2.9) tratam do cálculo do *bandwidth* da união e da junção de dois grafos com conjuntos de vértices disjuntos.

$$bw(G \cup H) = \max\{bw(G), bw(H)\} \tag{2.8}$$

A demonstração da equação (2.8) é bem simples, e será omitida, mas a demonstração da equação (2.9), não. Esta será apresentada na seção 4.4.

$$\begin{aligned} bw(G + H) = \min\{ \quad & \max\left\{bw(G), \left\lceil \frac{|V(G)|}{2} - 1 \right\rceil\right\} + V(H) \quad , \\ & \max\left\{bw(H), \left\lceil \frac{|V(H)|}{2} - 1 \right\rceil\right\} + V(G) \quad \} \end{aligned} \tag{2.9}$$

A relação entre o *bandwidth* e o tamanho da maior clique de um grafo é dada na inequação (2.10). Esta é uma consequência da inequação (2.6) em conjunto com a equação (2.2).

$$bw(G) \geq \omega(G) - 1 \tag{2.10}$$

Finalmente, apresentamos um limite que envolve os diâmetros dos subgrafos de um grafo.

Lema 2.2.

$$bw(G) \geq \max\left\{\left\lceil \frac{|V(H)| - 1}{diam(H)} \right\rceil : H \textit{ é subgrafo conexo de } G\right\}$$

Demonstração. Seja f uma rotulação ótima de $H \subseteq G$. Dois vértices adjacentes de H têm sempre rótulos com diferença menor ou igual a $bw_f(H)$. Então, dados dois vértices quaisquer u e v de H, considerando o menor caminho entre u e v, temos

que $f(v) \leq f(u) + d(u,v) \times bw_f(H)$, que implica

$$bw_f(H) \geq \frac{f(v) - f(u)}{d(u,v)}.$$

Considere o caso $f(u) = 1$ e $f(v) = n$. Então,

$$bw(H) \geq \frac{n-1}{d(u,v)}$$

e, como $diam(H) \geq d(u,v)$, tem-se

$$bw(H) \geq \frac{n-1}{d(u,v)} \geq \frac{n-1}{diam(H)}$$
$$bw(H) \geq \frac{n-1}{diam(H)}$$

Usando o resultado da equação (2.6), chega-se ao lema 2.2. □

2.4 *Pathwidth*

Além de *bandwidth*, outros parâmetros de largura em grafos têm sido estudados. Nesta seção, apresentamos os parâmetros *pathwidth* e *proper pathwidth*, e mostramos a relação que eles possuem com *bandwidth*. Em particular, que os valores de *bandwidth* e *proper pathwidth* coincidem em um mesmo grafo G.

Definição (Decomposição em caminho)**.** *Uma* ***decomposição em caminho****, ou* **path decomposition***, de um grafo G é um mapeamento de $G = (V, E)$ em um caminho $P = (F_V, W)$, sendo F_V uma família de subconjuntos de V, e W um conjunto de arestas com extremos em F_V, respeitando as seguintes regras:*

1. *a união de todos os conjuntos de F_V deve ser o conjunto dos vértices de G:*

$$\bigcup_{X \in F_V} X = V(G);$$

2. *para cada aresta uv de $E(G)$, os vértices u e v devem ambos pertencer a pelo menos um conjunto X de F_V:*

$$\forall uv \in E(G),\ \exists X \in F_V :\ u, v \in X;$$

3. *sejam X e Y dois conjuntos de F_V, e $v \in V(G)$; se v pertence a $X \cap Y$, então v pertence a todos os conjuntos Z que estão entre X e Y em P.*

Uma **decomposição em caminho própria** é uma decomposição em caminho que satisfaz a propriedade abaixo:

1. Para qualquer par de vértices u e v em $V(G)$, o conjunto $\{X : u \in X\}$ não é subconjunto próprio de $\{X : v \in X\}$.

A **largura** de uma decomposição em caminho $P = (F_V, W)$ é dada por

$$pw_P(G) = \max\{|X| - 1 : X \in F_V\}.$$

O valor do *pathwidth* de G é dado por

$$pw(G) = \min\{pw_P(G) : P \text{ é uma decomposição em caminho de } G\}$$

enquanto que o valor do *proper pathwidht* de G é dado por

$$ppw(G) = \min\{pw_P(G) : P \text{ é uma decomposição em caminho própria de } G\}$$

Teorema 2.1. *Todo grafo G (conexo) possui uma decomposição em caminho própria.*

Demonstração. Considere uma rotulação f como uma permutação dos vértices de G sobre uma reta numerada de 1 a n. Defini-se cada elemento X_{uv} da família F_V em função de cada aresta uv de G, de forma que X_{uv} seja o conjunto de vértices de G que estão entre o vértice u e o vértice v na reta de rotulação. Desconsidere (remova de F_V) os conjuntos X_{uv} que estejam totalmente contidos em outro conjunto de F_V. A ordem dos elementos de F_V no caminho C da decomposição deve ser crescente em função do menor rótulo contido em cada elemento de F_V. Será mostrado que essa definição de elementos de F_V satisfaz os requisitos de uma decomposição em caminho, e que $pw(G)$ correspondente a essa decomposição possui valor igual e $bw(G)$.

Vê-se que essa definição satisfaz o primeiro requisito da decomposição porque G conexo implica que todo vértice está associado a pelo menos uma aresta, e os

conjuntos desconsiderados têm seus elementos totalmente contidos em um outro conjunto de F_V.

O segundo requisito é claramente satisfeito porque os vértices de cada aresta uv de G estarão juntos no elemento X_{uv} de F_V por definição, e as arestas uv cujos conjuntos X_{uv} foram desconsiderados têm seus elementos totalmente contidos em um outro conjunto de F_V.

O terceiro requisito é demonstrado por contradição. Suponha que exista um vértice v que pertença a Y_i e a Y_j de F_V tal que $i < j$, e que v não pertença a pelo menos um Y_k, onde $i < k < j$. Considere sem perda de generalidade que Y_k seja o primeiro elemento do caminho para o qual v não pertença. Se $v \in Y_i$, então a aresta i inicia antes de v (se i inicia em v, então j obrigatoriamente inicia também em v, e assim ou Y_i é subconjunto próprio de Y_j ou o oposto: impossível), e termina ou nele ou depois dele. Se $v \in Y_j$, então a aresta j inicia ou em v ou antes dele, e termina depois dele. Como Y_k está entre Y_i e Y_j, por causa da ordem dada pelo elemento de menor rótulo no conjunto, então a aresta k inicia antes de v, mas não pode terminar depois dele, porque ocorreria de $v \in Y_k$. Logo, k termina antes de v. Entretanto, isso implica que a aresta k inicia depois de i e termina antes de i, que implica Y_k ser subconjunto próprio de Y_i: contradição, porque os subconjuntos próprios de outros conjuntos em F_V foram removidos da família por definição. □

Lema 2.3. *O proper pathwidth de um grafo G é*

$$ppw(G) = \min\{\omega(H) \ : H \textit{ supergrafo de intervalo próprio de } G\} - 1$$

Demonstração. Seja $T = (F_V, W)$ uma decomposição em caminho própria de G tal que $pw_T(G) = ppw(G) = k$. Como $pw_T(G) = k$, então $\max\{X_i \ : X_i \in F_V\} = k+1$. Será construído um supergrafo de intervalo próprio G' de G tal que $\omega(G') = k + 1$, logo $\min\{\omega(H) \ : H$ supergrafo de intervalo próprio de $G\} \leq k + 1$, ou seja, $\min\{\omega(H) \ : H$ supergrafo de intervalo próprio de $G\} - 1 \leq ppw(G)$.

De fato, seja $\{I_v\}_{v \in V(G)}$ o conjunto dos intervalos correspondentes aos vértices v de G. Considere uma reta rotulada de 1 a $|F_V|$ como referência para definição dos intervalos. Cada intervalo I_v inicia no menor rótulo a da reta tal que $v \in X_a$, e termina no maior rótulo b da reta tal que $v \in X_b$. Seja G' o grafo de intervalo próprio gerado por este modelo de intervalos. O grafo G' é de intervalo próprio porque T é uma decomposição em caminho própria, que, por regra, não contém

inclusão própria entre quaisquer dois conjuntos de F_V. E G' é supergrafo de G por conter um intervalo para cada vértice de G, além de dois vértices adjacentes u e v de G, que estão num mesmo conjunto X_i de F_V (pela propriedade 2 da definição de decomposição em caminho), terem seus respectivos intervalos I_u e I_v adjacentes.

Como o maior conjunto X_i de F_v possui $k+1$ vértices distintos, então $k+1$ intervalos foram gerados a partir desse conjunto, o que confere ao grafo de intervalo próprio uma clique de tamanho $k+1$, logo $\omega(G') = k+1$. Assim, $ppw(G) \geq \min\{\omega(H) : H \text{ supergrafo de intervalo próprio de } G\} - 1$.

Por outro lado, seja G' um supergrafo de intervalo próprio de G tal que $\omega(G')$ seja mínimo. Considere $\omega(G') = k+1$. Será construída uma decomposição em caminho própria T, a partir de G', tal que T seja uma decomposição em caminho própria de G. E será mostrado que $pw_T(G) = \omega(G') - 1$, logo $ppw(G) \leq pw_T(G) = \omega(G') - 1$. Consequentemente, $ppw(G) = \omega(G') - 1$.

Seja $\{I_v\}_{v \in V(G')}$ o modelo de intervalos de G'. Considere o modelo de intervalos na representação canônica, com os $2|\{I_v\}|$ extremos distintos na reta. Nomeie os extremos, da esquerda para direita, como $p_1, \ldots, p_{2|\{I_v\}|}$. Defina X_i como os intervalos que contém o ponto p_i neles. A família de conjuntos $\{X_i\}$ é uma decomposição em caminho própria de G porque

1. o requisito "1." é claramente satisfeito;

2. para o requisito "2.", cada aresta $uv \in G$ também está no supergrafo G', e a adjacência de $uv \in G'$ é representada pela interseção de dois intervalos, que possuem pelo menos um ponto comum p_j, tal que $u, v \in X_j$;

3. no requisito "3.", se v está em X_i e em X_j, então está também nos conjuntos X_k tal que $i < k < j$ porque o intervalo I_v é contínuo;

4. o requisito "4." é satisfeito naturalmente em virtude de G' ser um grafo de intervalo próprio.

Como o maior $|X_i|$ é do tamanho da maior clique de G', por construção, então $pw_T(G) = \omega(G') - 1$. □

Agora, podemos apresentar a demonstração do principal resultado que relaciona estes parâmetros: o *bandwidth* de um grafo é igual ao seu *proper pathwidth*.

Teorema 2.2. *Para todo grafo G, $bw(G) = ppw(G)$.*

Demonstração. Seja G um grafo tal que $ppw(G) = k$. Considere G' um supergrafo de intervalo próprio de G, como descrito no lema 2.3. Desta forma, $\omega(G') = k + 1$, e será construída uma rotulação para G em função de um modelo de intervalos de G', como segue.

Seja $\{I_v\}_{v \in G'}$ um modelo de intervalos de G' com todos os extremos de todos intervalos distintos, sendo o extremo esquerdo de cada intervalo I_v rotulado com um valor inteiro – $p_{e(v)}$ – de 1 a $|V(G')|$ de acordo com a posição do extremo em relação ao modelo: o primeiro extremo esquerdo de um intervalo recebe rótulo 1, o segundo, 2, e assim em diante.

Define-se uma rotulação f para G como $f(v) = p_{e(v)}$ para todo $v \in G$. Suponha por contradição que existam, segundo essa rotulação, dois vértices adjacentes x e y de G tal que $|f(y) - f(x)| > k$. Isso implica que um corte na posição de rótulo $f(y)$ encontra, pelo menos, $k + 2$ intervalos: $f^{-1}(x), f^{-1}(x + 1), \ldots, f^{-1}(x + k + 1)$ (ver figura 2.4), em virtude de apenas o rótulo esquerdo dos intervalos serem numerados e dos intervalos não serem propriamente incluídos em outros. Mas isso é um absurdo porque $\omega(G') = k + 1$, logo $|f(y) - f(x)| \leq k$.

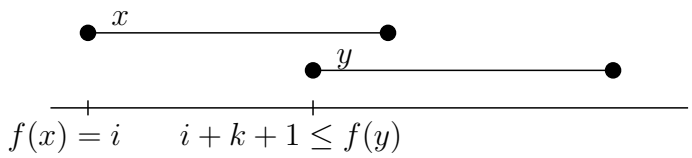

Figura 2.4: Modelo de intervalos de G' para x e y adjacentes.

Seja f uma rotulação ótima de G, e seja $\{I_v\}$ um conjunto de $|V(G)|$ intervalos de mesmo tamanho, com k unidades cada. Considere G' o grafo de intervalo representado pelo modelo de intervalos de $\{I_v\}$ tal que o extremo esquerdo de cada intervalo I_v está na posição $f(v)$ da reta. O grafo G' é um supergrafo de G pois, se $u, v \in V(G)$ são adjacentes, então $f(v) \leq f(u) + k$, e os intervalos I_u e I_v se intersectam. Além disso, como os intervalos $\{I_v\}$ são de mesmo tamanho, o grafo G' é um grafo de intervalo próprio. Pelo lema 2.3, $\omega(G') - 1 \geq k$, que implica $ppw(G) \geq bw(G)$. Conclui-se que $ppw(G) = bw(G)$. □

Capítulo 3

Demonstração de NP-completude

Neste capítulo são apresentadas, em detalhes, duas provas de que o problema BANDWIDTH é NP-completo.

BANDWIDTH foi mostrado ser NP-completo, pela primeira vez, em 1976, por Papadimitriou [29]. Posteriormente, Garey *et al* [6] mostraram que o problema permanece NP-completo para classes bem restritas de grafos; até mesmo em árvores com grau máximo 3. Outros resultados foram obtidos nos anos seguintes e podem-se listar [14], [5], [20], como os principais. Para algumas classes conhecidas o problema torna-se NP-completo por conterem uma subclasse de grafos para a qual o problema é NP-completo. E por esse fato não é necessário uma demonstração explícita de NP-completude. A tabela 2.2 sintetiza esses resultados.

3.1 Classe geral

A demonstração de que BANDWIDTH é NP-completo para a classe de grafos em geral foi feita num artigo de Papadimitriou [29] intitulado "*The NP-Completeness of the Bandwidth Minimization Problem*", publicado em 1976. Nele, uma série de transformações polinomiais são feitas: uma do problema AT LEAST 3SAT para o problema EXACTLY 3SAT, uma do problema EXACTLY 3SAT para o problema LAP, uma do problema EXACTLY 3SAT para o problema RESTRICTED LAP e uma de RESTRICTED LAP para BANDWIDTH. O problema AT LEAST 3SAT foi originalmente apresentado ser NP-completo no artigo clássico de Karp de 1972 [18].

Esta seção apresentada a demonstração de Papadimitriou [29]. Os problemas considerados usados são definidos a seguir.

Problema: EXACTLY 3-SATISFIABILITY PROBLEM — E3SAT

Instância: Uma família $F = \{F_1, \ldots, F_r\}$ de cláusulas sobre um conjunto $U = \{x_1, \ldots, x_n\}$ de variáveis tal que cada cláusula tem exatamente três literais.

Questão: Existe uma atribuição de verdade $t : U \to \{verdadeiro, falso\}$ que satisfaz todas as cláusulas de F?

Problema: LINEAR ARRAY PROBLEM — LAP

Instância: Um grafo $G = (V, E)$ e uma rotulação das arestas $f : E \to \mathbb{N}$.

Questão: Existe uma rotulação $g : V \to \{1, \ldots, n\}$ tal que $uv \in E$ implica $|g(u) - g(v)| \leq f(uv)$?

Problema: RESTRICTED LAP — RLAP

Instância: Um grafo $G = (V, E)$ e uma rotulação $f : E \to \{b, 2b - 1\}$, $b \in \mathbb{N}$.

Questão: Existe uma rotulação $g : V \to \{1, \ldots, n\}$ tal que $uv \in E$ implica $|g(u) - g(v)| \leq f(uv)$?

A primeira redução, AT MOST 3SAT $\propto$ E3SAT, não será apresentada, em virtude dela já ser bem conhecida na literatura, podendo ser encontrada, por exemplo, nos livros de Cormen *et al* [2] e Garey e Johnson [7].

A sequência de reduções no artigo de Papadimitriou é feita na seguinte ordem.

AT MOST 3SAT $\propto$ E3SAT $\propto$ LAP

e depois

E3SAT $\propto$ RLAP $\propto$ BANDWIDTH

Claramente, o problema BANDWIDTH é uma versão restrita de RLAP — por conter o mesmo valor de rótulo para todas as arestas —, que por sua vez é uma versão retrita de LAP. Se fosse apresentada unicamente uma redução de E3SAT para BW, os problemas LAP e RLAP seriam automaticamente mostrados NP-completos. Mas Papadimitriou, em vez da redução direta, optou por apresentar a cadeia de reduções.

O teorema 3.1 apresenta a demonstração de E3SAT $\propto$ LAP e o lema 3.1 a transformação de RLAP $\propto$ BANDWIDTH.

A redução de E3SAT $\propto$ RLAP não é feita diretamente. O artigo explica como deve ser feita a alteração no grafo G de LAP para o grafo G' de RLAP de forma que o teorema 3.1 possa ser reaplicado para a redução de E3SAT para RLAP, usando as mesmas estruturas e técnicas de demonstração, adaptando apenas algumas características e os rótulos das arestas. Tais alterações estão apresentadas no corolário 3.1.

Teorema 3.1. LAP *é NP-completo.*

Demonstração. A demonstração será feita pela redução a partir de E3SAT. Seja uma instância B deste problema com as variáveis $x_1, \ldots, x_n$ e cláusulas $F_1, \ldots, F_r$. Define-se S como o conjunto de literais do problema: $S = \{x_i, \overline{x}_i : i = 1, \ldots, n\}$. É necessário construir um grafo $G = (V, E)$ e uma função de rotulação das arestas f tal que B é satisfatível se, e somente se, G e f têm solução para o problema LAP. Será usada a nomenclatura $LAP(G)$ para representar a expressão "G e f têm solução para o problema LAP".

Um rótulo para uma aresta uv é definido no contexto do problema LAP como o valor máximo do módulo da diferença dos rótulos de u e v. E uma reta de rotulação é uma nomenclatura para a rotulação dos vértices do grafo G em analogia aos vértices serem rotulados ao distribuí-los sobre uma reta numerada de 1 a n; nesse caso, o rótulo do vértice é a posição na reta que ele ocupa.

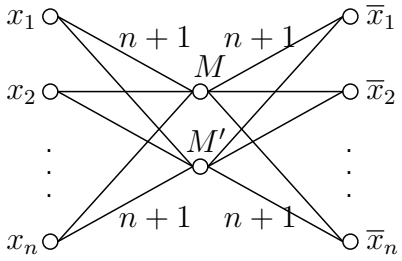

Figura 3.1: Grafo usado para dividir o conjunto de literais de S em dois conjuntos P e Q de tamanho n cada. Todas as arestas têm rótulo $n + 1$.

Considere o grafo H da figura 3.1, que será usado na construção do grafo final. Ao distribuir os $2n + 2$ vértices de H numa reta de rotulação de tal forma que as posições (rótulos) dos vértices respeitem as restrições de distância das arestas, os vértices M e M' devem ocupar as posições $n + 1$ e $n + 2$, no centro; caso contrário, algum dos literais, representados por $x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n$, estará a uma distância de M ou M' maior que $n + 1$. As posições de M e M' implicam uma partição do conjunto S em dois conjuntos P e Q tal que $|P| = |Q| = n$.

O grafo final G irá conter $n + r + 1$ instâncias de H $(H_1, \ldots, H_{n+r+1})$ e $n + r$ vértices A_i, $1 \leq i \leq n+r$. Cada A_i é unido aos vértices M e M' tanto de H_i quanto de H_{i+1} por arestas de rótulo $n+2$. Uma solução para LAP com a estrutura descrita até o momento pode ser obtida ao colocar H_1, seguido de A_1, e depois H_2, seguido de A_2, repetindo esse padrão de organização até estabelecer o último H_{n+r+1}. E cada H_i internamente deve ser organizado para respeitar a divisão de literais em igual quantidade em torno de M e M' centrais.

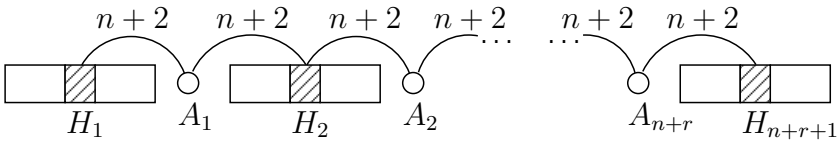

Figura 3.2: Grafo G construído a partir da instância B.

Seguindo a construção de G, conectam-se os literais correspondentes (todos os x_1 de todas as cópias, por exemplo) das cópias de H com arestas de rótulo $2n + 5$. Isso obriga as $n+r+1$ cópias de H a conterem a mesma divisão de literais em torno dos vértices centrais.

Para cada valor de i $(1 \leq i \leq n)$, liga-se A_i a x_i e a $\overline{x}_i$ em H_i com arestas de rótulo $n + 3$. Nessa configuração, as primeiras n cópias de H forçam a partição $S = P + Q$ ser consistente, ou seja, impede que x_i e $\overline{x}_i$, qualquer que seja i, estejam juntos ou em P ou, consequentemente por $|P| = |Q|$, em Q. Se por contradição x_i e $\overline{x}_i$ estiverem juntos em P (sem perda de generalidade) para algum valor de i, então A_i estará distante pelo menos $n + 4$ unidades ou de x_i ou de $\overline{x}_i$, que não é uma solução aceitável para LAP. Ver figura 3.3.

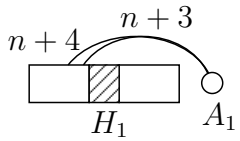

Figura 3.3: Caso em que os literais x_i e $\overline{x}_i$ estão ambos em P, ocasionando uma aresta com distância $n + 4$.

Nas próximas r cópias de H em G, a partição $P + Q$ é forçada a satisfazer B. Note que $S = P + Q$ induz uma atribuição de verdade t às variáveis do problema ao definir verdadeiro ou falso a todos os literais de um mesmo conjunto; seja t sem perda de generalidade a atribuição que todo literal de P tem valor falso, ao passo

que todo literal de Q tem valor verdadeiro. Liga-se o vértice A_{n+i} aos três literais da cláusula F_i na cópia H_{i+n} com arestas rotuladas de $n+4$. Se t não satisfizer B, então haverá uma cláusula F_i na qual todos os três literais terão valor falso, ou seja, pertencerão a P; porém, sendo A_i ligado a três literais em P por arestas de tamanho $n+4$, ocorrerá obrigatoriamente ao menos um dos literais com distância maior ou igual a $n+5$ de A_i, que não é uma solução de LAP.

Por outro lado, se t é uma atribuição de verdade que satisfaz B, então a partição $S = P + Q$ é encontrada ao colocar os literais negativos em P e os positivos em Q. Entretanto, essa distribuição dos literais não é suficiente para que LAP seja resolvido; é necessário ainda apresentar uma ordenação dos literais nos conjuntos P e Q de forma que as restrições de rotulação das arestas sejam satisfeitas. Essa ordenação será apresentada nos parágrafos seguintes.

Observa-se que as primeiras n cópias de H possuem, em cada cópia, duas arestas que restringem a colocação dos literais: $2n+5$ e $n+3$ (além dos rótulos $n+1$ que estão presentes em todas as cópias de H). E as r cópias seguintes possuem cada uma duas arestas que restringem as posições dos literais: $2n+5$ e $n+4$.

Serão analisadas inicialmente as últimas cópias de H que envolvem as arestas $2n+5$ e $n+4$. Seja t uma atribuição de verdade para as variáveis de B em que $P = \{\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}\}$ seja uma ordem qualquer dos literais negativos, e $Q = \{x_1, x_2, \ldots, x_n\}$ a ordem dos positivos em função de P. Sabe-se que na cópia H_k a cláusula F_k deverá ser representada, e que no máximo dois literais de F_k são negativos. Iniciando na última cópia, H_{n+r}, a disposição dos literais negativos deve ser gerada ao mover em P os literais negativos de F_r (se houver) para a direita. Se não houve literais negativos, então a disposição dos literais permane como inicialmente. Por exemplo, se, para uma instância de E3SAT com 5 cláusulas e 8 variáveis, a cláusula F_5 for $(\overline{x_4}, x_7, \overline{x_8})$, e uma solução do problema for a atribuição de verdade $t(x_1, \ldots, x_8) = true$, então a disposição inicial do literais de P em H_5 será $P = \{\overline{x_1}, \overline{x_2}, \overline{x_3}, \overline{x_5}, \overline{x_6}, \overline{x_7}, \overline{x_4}, \overline{x_8}\}$.

Seguinte na ordem descrecente das cláusulas, de F_{r-1} para F_1, seja F_k a cláusula em análise. Deve-se gerar a ordem dos literais negativos de H_k ao copiar a ordem dos literais negativos de H_{k+1} e mover os literais negativos de F_k para a direita. Esta ordenação garante que a restrição das duas arestas com rótulos $2n+5$ e $n+4$ serão satisfeitas. Essa ordenação deve ser repetida até a análise da cláusula F_1.

Serão analisadas agora as n primeiras cópias de H que envolvem as arestas $2n+5$

e $n+3$. Para esse conjunto de cópias, a mesma regra de ordenação anterior deve ser usada, movendo neste caso apenas o literal negativo de H_k com aresta $n+3$, $1 \leq k \leq n$, para a direita a partir da ordenação de P da cópia H_{k+1}.

Dessa forma, B é satisfatível se, e somente se, $LAP(G)$. □

Corolário 3.1. RLAP *é NP-completo.*

Demonstração. Na demonstração do teorema 3.1, há algumas mudanças que podem ser feitas na construção do grafo G, para que o novo grafo G' resultante possua apenas dois valores para os rótulos das arestas. Isso permite que a mesma demonstração do teorema seja usada, com as necessárias adaptações, para provar que RLAP é NP-completo. Essa é a abordagem usada nesta demonstração.

Define-se o grafo K_p como o completo de p vértices, cujas arestas têm comprimento $p-1$. Note que se K_p for subgrafo de G' com p sendo maior que os rótulos de todas as arestas de G', então os vértices de K_p irão ocupar obrigatoriamente as primeiras ou as últimas p posições da rotulação numa solução de $LAP(G')$. Dois grafos K_p serão incluídos em G (para formar G'): um deles, substituindo a cópia H_{n+r+1} de G; o outro, apenas acrescentado ao grafo.

As arestas que interligam os grafos passam a ser apresentadas a seguir. O ponto chave para entender a solução nesta nova organização é que G', com dois subgrafos K_p, terá as posições dos blocos M de cada cópia H_i e dos blocos A_i precisamente definidas tanto pela obrigatoriedade dos K_p estarem nos extremos quanto pelo valor dos rótulos das arestas. Em G', por exemplo, serão usadas as duas estruturas das figuras 3.1 e 3.4 para construir as cópias de H, sendo que a nova estrutura, da figura 3.4, porém, não obriga sozinha que o conjunto de literais seja dividido igualmente com $|P| = |Q|$.

Como dito no parágrafo anterior, em vez de dois vértices M e M' no grafo H, um bloco de $m_i \geq 2$ vértices vai ser usado; o grafo H_i passa a ser considerado um bloco de m_i vértices centrais ligados aos literais x_i e $\overline{x}_i$. As duas opções de m_i estão apresentadas nas figuras 3.1 e 3.4. E, em vez de um vértice A_i, usa-se um bloco de a_i vértices.

O grafo G' é composto por duas cópias de K_p, $n+r$ cópias de H e $n+r-1$ cópias de A. A quantidade de vértices em cada bloco M das cópias de H e de vértices nos blocos A está apresentada na tabela 3.1.

A_{n+r} é na realidade composto por parte dos vértices de um dos K_p do G'.

	$1 \leq i \leq n-1$	$i = n$	$n+1 \leq i \leq n+r$
m_i	3	3	2
a_i	2	3	3

Tabela 3.1: Quantidade de vértices em cada bloco do grafo G'.

- O grafo K_p que será rotulado no início da solução de $LAP(G')$ terá os vértices v_{p-1} e v_p ligados aos vértices do bloco M de H_1 com arestas de rótulo $n+4$.
- Para $1 \leq i \leq n-1$, o bloco A_i terá os dois vértices ligados ao bloco M de H_i e ao bloco M de H_{i+1}; todas as arestas com rótulos $n+4$.
- O bloco A_n tem três vértices: dois deles ligados a todos os três vértices do bloco M de H_n, e o terceiro ligado a apenas dois dos vértices do bloco M de H_n; e também todos os vértices de A_n ligados ao bloco M de H_{n+1}; todas as arestas com rótulo $n+4$.
- Para $n+1 \leq i \leq n+r-1$, o bloco A_i terá os três vértices ligados ao bloco M de H_i e ao bloco M de H_{i+1}; todas as arestas com rótulos $n+4$.
- O grafo K_p que será rotulado no fim da solução de $LAP(G')$ terá os vértices v_1, v_2 e v_3 ligados aos vértices do bloco M de H_{n+r} com arestas de rótulo $n+4$.
- As primeiras n cópias de H forçam a partição de S ser consistente, e esse objetivo é alcançado ao ligar, para $1 \leq i \leq n$, um dos vértices de A_i aos literais x_i e $\overline{x_i}$ na cópia H_i com aresta de tamanho $n+4$. No caso $i = n$, o vértice que será escolhido deve ser um dos dois que foram ligados ao bloco M de H_n.
- As r cópias seguintes de H forçam a solução de B ser satisfatível devem ligar, para $n+1 \leq i \leq n+r$, um dos vértices de A_i aos três literais da cláusula F_i com arestas de tamanho $n+4$. No caso $i = n+r$, deve-se usar um dos vértices v_1, v_2 ou v_3 de K_p para fazer a ligação.
- Para que as partições encontradas nas $n+r$ cópias de H sejam iguais, os literais x_j das cópias consecutivas de H devem ser ligados com arestas de rótulo $2n+7$.

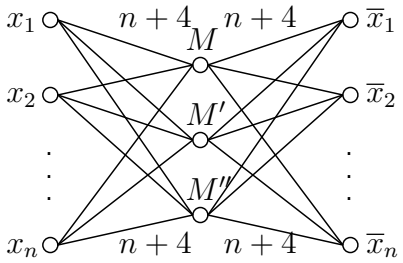

Figura 3.4: Grafo usado para dividir o conjunto de literais de S em dois conjuntos P e Q de tamanho n cada. Todas as arestas têm rótulo $n+4$.

O grafo resultante G', construído de acordo com os parâmetros acima, possui arestas de valores $n+4$ e $2n+7$, que produzem uma instância de grafo com rótulos de dois valores b' e $2b'-1$, que é uma instância do problema RLAP. □

Além do grafo K_p definido na demonstração do corolário 3.1, define-se K_p^q para $p \leq q$ como o grafo de vértices $\{v_1, \ldots, v_q\}$ e arestas $\{v_i v_j : |i-j| \leq p\}$, com todas as arestas rotuladas com p. Assim como K_p, se K_p^q for subgrafo de algum grafo G, e q for maior que todos os rótulos de arestas de G, então K_p^q será rotulado ou com os valores iniciais ou com os finais de uma rotulação, em qualquer solução de $LAP(G)$. As arestas com rótulo p garantem que os vértices de K_p^q devem ser rotulados consecutivamente e os q vértices, tal que $q \geq \{\text{MAIOR RÓTULO DAS ARESTAS}\}$, garantem que o grafo não pode ter uma aresta externa atravessando todos os vértice, logo K_p^q deve ser rotulado num dos extremos.

Lema 3.1. *Qualquer instância de* RLAP *com* $m > 0$ *arestas de rótulo* $2b-1$ *pode ser transformada em tempo polinomial numa instância do mesmo problema com* $m-1$ *arestas de rótulo* $2b'-1$*, onde* b' *é o parâmetro da nova instância de* RLAP*.*

Demonstração. Seja $G = (V, E)$ um grafo com a rotulação das arestas dada pela função l, e seja uv uma aresta de G com $l(uv) = 2b-1$, com $b \in \mathbb{Z}$. E seja o inteiro $k = \lceil \frac{n}{b} \rceil$.

O grafo G será transformado no grafo G' pelos seguintes passos.

1. Adicione um novo vértice c a G, e as arestas uc e vc com rótulo $b+1$. Remova a aresta uv.

2. Incremente o valor do rótulo de todas as arestas de G de b para $b+1$, e de $2b-1$ para $2b+1$.

3. Adicione duas cópias do grafo K_{b+1}^{2b+1} com vértices $\{v_1, v_2, \ldots\}$ e $\{v'_1, v'_2, \ldots\}$, respectivamente.

4. Adicione as cadeias

$$C = \{c = c_0, c_1, \ldots, c_k = v_1\}, C' = \{c = c'_0, c'_1, \ldots, c'_k = v'_1\}$$

com arestas $c_{i-1}c_i$ e $c'_{i-1}c'_i$ com rótulo $b+1$ para $i = \{1, \ldots, k\}$.

O resultado da transformação está esboçado na figura 3.5.

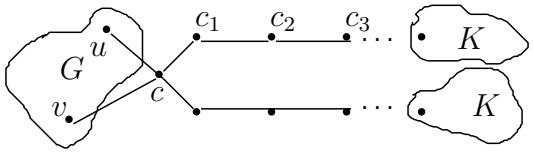

Figura 3.5: Construção de G' a partir de G.

Suponha que $LAP(G')$ tenha solução. Pela característica dos grafos K_{b+1}^{2b+1}, eles serão colocados no início e no fim da reta de rotulação. E pelo rótulo das arestas de G', a cada intervalo na reta de $b+1$ unidades, deve haver pelo menos um vértices das cadeias C e C' adicionadas ao grafo. A solução de G a partir da rotulação de G' é então remover os extremos K_{b+1}^{2b+1} e os vértices das cadeias. Os vértices resultantes, todos de G apenas, serão mantidos na ordenação da solução de G', mas os rótulos serão remarcados de acordo com a nova posição em relação ao início da reta de rotulação.

Agora suponha que $LAP(G)$ tem solução. Sabe-se que no pior caso a aresta uv tem seus vértices distantes $2b-1$ unidades um do outro. Então a solução de $LAP(G')$ é obtida colocando-se c no meio entre u e v, e ordenando os outros vértices c_i e c'_i das cadeias a cada $b+1$ unidades de distância, até que todo o grafo G seja atravessado. As cópias de K_{b+1}^{2b+1} são colocadas nos extremos da rotulação.

Conclui-se que $LAP(G')$ tem solução se, e somente se, $LAP(G)$ tem solução. Ademais, o grafo G' pode ser construído a partir de G em tempo polinomial, e tem $m-1$ arestas de rótulo $2b+1$. □

Teorema 3.2. BANDWIDTH *é NP-completo.*

Demonstração. Reduzindo RLAP até BANDWIDTH, ao executar m vezes a transformação do lema 3.1 na instância de RLAP, até que não exista aresta com rótulo igual a $2b - 1$. □

3.2 Cobipartido e *split*

A demonstração de NP-completude do problema BANDWIDTH para a classe dos grafos *split* é feita pela redução a partir deste mesmo problema para a classe dos grafos cobipartidos, que foi provado ser NP-completo por Kloks, Kratsch e Müller em 1998 [22].

Definição (Grafo *split*)**.** *Um grafo é split se o seu conjunto de vértices puder ser particionado em um conjunto independente e uma clique.*

Definição (Grafo cobipartido)**.** *Um grafo G é cobipartido se, e somente se, o complemento de G ($\overline{G}$) for bipartido.*

Lema 3.2. *Todo grafo cobipartido $G = (X, Y, E)$ possui uma rotulação ótima f tal que $f(x) < f(y)$ para todo $x \in X$ e $y \in Y$.*

Demonstração. Seja f uma rotulação ótima de G e considere $|X| + |Y| = n$. Se $f^{-1}(1)$ e $f^{-1}(n) \in X$, então, pelo lema 2.1, G é um grafo completo. Analogamente, se $f^{-1}(1)$ e $f^{-1}(n) \in Y$, então G também é um grafo completo. Nesses casos, qualquer rotulação dos vértices implica $bw(G) = n-1$; em particular, uma rotulação na qual todos os vértices de X são rotulados antes dos vértices de Y, que satisfaz a afirmação do lema.

Considere, então, que $f^{-1}(1) \in X$ e $f^{-1}(n) \in Y$. Será mostrado que é possível transformar f numa rotulação g em que $g(x) < g(y)$ para todo $x \in X$ e $y \in Y$ tal que $bw_g(G) \leq bw_f(G)$.

Considere que existe um par de vértices $x \in X$ e $y \in Y$ tal que $f(y) < f(x)$. Seja g a seguinte rotulação.

$$g(w) = \begin{cases} f(x) & \text{se } w = y \\ f(y) & \text{se } w = x \\ f(w) & \text{se } w \neq x, y \end{cases}$$

A figura 3.6 apresenta a troca de rótulos entre x e y na rotulação g. Os dois fatos a seguir podem ser observados.

Fato 1. Qualquer vértice z adjacente a x com $f(z) < f(y)$ terá, em g, $|g(x) - g(z)| \leq |f(x) - f(z)| \leq bw_f(G)$.

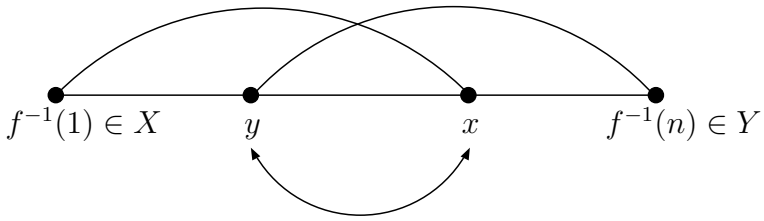

Figura 3.6: Exemplificação da troca de rótulos entre os vértices x e y.

Fato 2. Qualquer vértice z adjacente a x com $f(z) > f(y)$ terá, em g, $|g(z) - g(x)| \leq |n - f(y)| \leq bw_f(G)$.

Os mesmos fatos podem ser analogamente observados para y, logo o $bw_g(G)$ não é maior que $bw_f(G)$ pela inversão dos rótulos dos vértices. Então, repetindo esse procedimento para todo par de vértices x e y com $f(y) < f(x)$, consegue-se uma rotulação g com $bw_g(G) \leq bw_f(G)$ tal que $f(x) < f(y)$ para todos $x \in X$ e $y \in Y$. Como $bw_f(G)$ é ótima, então $bw_g(G) = bw(G)$. □

Teorema 3.3 (Kloks *et al* [20])**.** BANDWIDTH *é NP-completo para a classe dos grafos split.*

Demonstração. A demonstração será feita pela redução a partir de BANDWIDTH para cobipartidos.

Seja $G = (X, Y, E)$, $|Y| \leq |X|$, um grafo cobipartido com $bw(G) \leq k$. Construa um grafo *split* H obtido de G como segue: adicione a G dois novos conjuntos de vértices A e B tais que $|A| = |X|$ e $|B| = k + 1 - |X|$ (garante-se que $|B| \geq 0$ em virtude da inequação (2.10)). O conjunto $A \cup Y$ forma um conjunto independente (em H, remova as arestas uv, $u, v \in Y$) e o conjunto $B \cup X$ forma uma clique (em H, adicione as arestas $uv, u \in B, v \in B \cup X$). Adicione também as arestas uv tal que $u \in A$ e $v \in B$.

Uma rotulação f de G tal que $bw_f(G) \leq k$ como descrita no lema 3.2 será usada para criar uma rotulação g de H tal que $bw_g(H) \leq k$. Os rótulos em g serão dados aos vértices de H na ordem $ABXY$, ou seja, uma rotulação dos vértices de H na qual $g(a) < g(b) < g(x) < g(y)$ para todo $a \in A$, $b \in B$, $x \in X$ e $y \in Y$; a rotulação dos vértices de X e Y, em especial, é dada na mesma ordem da rotulação f. Como $|A \cup B| = k + 1$, $|B \cup X| = k + 1$, $bw_f(X \cup Y) \leq k$, e não há arestas entre A e X, A e Y, e B e Y, então não há diferença de rotulação maior que k em g, logo $bw_g(H) \leq k$.

Como $|A \cup B| = |B \cup X| = k+1$, e $G[B \cup X]$ é uma clique, a única possibilidade de uma rotulação g de H com $bw_g(H) \leq k$ é uma rotulação na qual os rótulos são dados na ordem $ABXY$ ou na ordem simétrica $YXBA$. Além disso, as arestas internas em Y, que existem em G mas não em H, não implicam uma rotulação g com $bw_g(G) > k$, porque $|Y| \leq |X| < k$ por hipótese, logo $bw_g(H) \leq k$, e consequentemente $bw_f(G) \leq k$, considerando que a rotulação f nesse caso seja a rotulação dos vértices dos conjuntos X e Y apenas.

Observe que se H for desconexo, os vértices do conjunto Y podem ser rotulados antes ou depois da rotulação ABX, mas é possível restringi-los ao lado direito da rotulação, sem prejuízo ao resultado obtido. □

Capítulo 4

Algoritmos polinomiais

Apenas para algumas classes existe algoritmo polinomial para resolver o problema BANDWIDTH. Neste capítulo, são apresentados os algoritmos para algumas classes: *caterpillar* com cabelos de tamanho no máximo 2, intervalo, *quasi-threshold*, cografo, P_4-esparso e *chain*.

Para a classe de intervalo, existem dois algoritmos polinomais: um $O(nk)$ e outro $O(n \log n)$, em tempo, sendo n o número de vértices e k o valor de entrada do algoritmo para o qual se quer verificar se o grafo admite uma rotulação f tal que $bw_f(G) \leq k$. Para as outras classes, conhece-se apenas um algoritmo.

4.1 *Caterpillar* com cabelos de tamanho no máximo 2

A classe de grafos *caterpillar* com cabelos de tamanho no máximo 2 possui um algoritmo polinomial $O(n \log n)$ para resolver o problema BANDWIDTH.

Em 1981, Assmann, Peck, Syslo e Zak [1] apresentaram um algoritmo de rotulação de um grafo *caterpillar* G em que um valor k de $bw(G)$ é testado. Se $bw(G) \leq k$, então o algoritmo consegue uma rotulação f de G tal que $bw_f(G) = k$; caso $bw(G) > k$, é demonstrado no artigo que o algoritmo será interrompido antes de rotular todos os vértices do grafo e que, nesse caso, existe um subgrafo de H de G tal que $bw(H) > k$. Para que seja encontrado o valor de $bw(G)$, é necessário usar esse algoritmo de rotulação e fazer uma busca sobre os valores possíveis de $bw(G)$: de 1 a n.

Interessante que, em 1986, Monien [28] apresentou uma demonstração de NP-completude do problema BANDWIDTH para grafos *caterpillar* com cabelos de tamanho no máximo 3, estabelecendo para essa classe de grafos um limite preciso onde o problema torna-se NP-completo.

Definição (*Caterpillar* com cabelos de tamanho no máximo p)**.** *Uma caterpillar com cabelos de tamanho no máximo p é uma árvore $G = (V, E)$ em que o subgrafo induzido pelo conjunto dos vértices de grau maior ou igual a 3 está contido em um caminho C de G, chamado corpo, e cada caminho C_i de $G[V \setminus V(C)]$ tem tamanho no máximo $p-1$. Seja v_i o vértice de C adjacente a algum vértice de C_i; $G[V(C_i) \cup \{v_i\}]$ é chamado cabelo.*

A figura 4.1 exemplifica um grafo *caterpillar*, e a figura 4.2 exemplifica um grafo *caterpillar* com cabelos de tamanho no máximo 2.

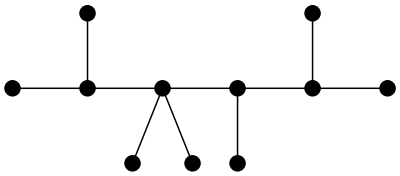

Figura 4.1: Exemplo de grafo *caterpillar*.

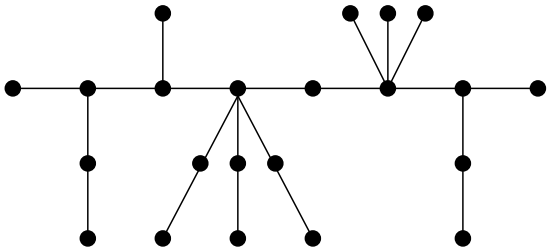

Figura 4.2: Exemplo de grafo *caterpillar* com cabelos de tamanho 1 e 2.

O algoritmo 1 resolve o problema BANDWIDTH para um grafo *caterpillar* G com cabelos de tamanho no máximo 2, em tempo $O(n \log n)$, encontrando uma rotulação f para G tal que $bw_f(G) \leq k$ sempre que G permitir uma rotulação dessa forma, e retorna verdadeiro nesse caso. Se $bw(G) > k$, então o algoritmo retorna falso.

No algoritmo 1, será usada a notação apresentada a seguir. A letra d representa o tamanho do caminho principal. Os vértices do corpo serão chamados de pontos,

Algoritmo 1 Seja G um grafo *caterpillar* com cabelos de tamanho no máximo 2. Encontrar uma rotulação f de G tal que $bw_f(G) = k$, em tempo $O(n)$.

Entrada: Um grafo $G = (V, E)$ *caterpillar* com cabelos de tamanho no máximo 2 e $k \in \mathbb{N}$

Saída: **verdadeiro** $\Leftrightarrow bw(G) \leq k$

1: Rotular os pontos com $0, k, \ldots, dk$
2: {No laço **para** a seguir, os cabelos dos ponto ik devem ser rotulados dando a cada vértice o menor rótulo possível tal que $bw_f(G) \leq k$. Se tal rotulação não puder ser feita, então o algoritmo deve retornar falso}
3: **para** $i = 1$ to $d - 1$ **faça**
4: Rotular o maior número possível de cabelos de comprimento 2 do ponto ik tal que aos vértices extremos sejam dados rótulos entre $(i-2)k$ e $(i-1)k$
5: Rotular todos os cabelos de comprimento 1, cujos rótulos devem ser $\geq (i-1)k$
6: Rotular o maior número possível de cabelos de comprimento 2 do ponto ik tal que ambos os vértices recebam rótulos entre $(i-1)k$ e ik
7: Rotular todos os vértices distantes uma unidade do ponto ik
8: Rotular todos os vértices restantes distantes 2 unidades do ponto ik, na mesma ordem dos rótulos dados no passo anterior
9: **fim para**

para facilitar a escrita e o entendimento do texto, além de se poder referenciar um determinado ponto pelo seu respectivo rótulo, ao usar a expressão "ponto ik".

A figura 4.3 exemplifica a rotulação dada pelo algoritmo 1 a um grafo *caterpillar* de cabelos com tamanho no máximo 2. A ideia subjacente da rotulação é que não é dado, a nenhum vértice distante duas unidades de um ponto ik, um rótulo entre $(i-1)k$ e $(i+1)k$, a menos que (na linha 8 do algoritmo) não exista vértice distante uma unidade que ainda precise ser rotulado; ou (como na linha 6) seja opcional fazer essa escolha. É importante observar no algoritmo que nenhum cabelo anexado a um ponto ik do grafo é rotulado com rótulo entre ik e $(i+1)k$ a menos que todos os rótulos entre $(i-1)k$ e ik sejam usados; e que nenhum rótulo entre $(i+1)k$ e $(i+2)k$ é usado a menos que todos os rótulos entre ik e $(i+1)k$ sejam usados .

É importante notar neste momento que, durante a rotulação de um ponto ik, os rótulos entre ik e $(i+1)k$ só são usados quando todos os rótulos entre $(i-1)k$ e ik já tiverem sido usados. E nenhum rótulo entre $(i+1)k$ e $(i+2)k$ é usado antes que todos os rótulos entre ik e $(i+1)k$ tenham sido usados.

A demonstração do teorema a seguir usa o resultado do lema 2.2, que foi apresentado e demonstrado na seção 2.3.

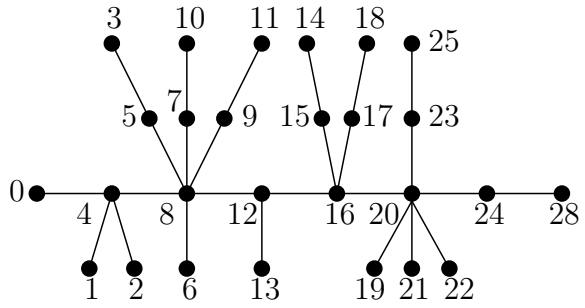

Figura 4.3: Exemplo de rotulação de um grafo *caterpillar* de cabelos com tamanho no máximo 2.

Teorema 4.1. *Seja G um grafo caterpillar com cabelos de tamanho no máximo 2, e seja $k \in \mathbb{N}$. Se a rotulação f dada a G pelo algoritmo 1 não for $bw_f(G) = k$, então $bw(G) > k$.*

Demonstração. Obviamente, $bw_f(G) \not< k$ porque os dois primeiros pontos recebem rótulos 0 e k.

Seja o caso em que o algoritmo retorna falso; será mostrado que isso implica a existência de um subgrafo *caterpillar* G' de G, onde $\left\lceil \frac{|V(G')|-1}{diam(G')} \right\rceil > k$. Logo, pelo lema 2.2, $bw(G') > k$. Como $bw(G) \geq bw(G') > k$, então $bw(G) > k$. Por outro lado, se o algoritmo retorna verdadeiro, então $bw(G) = k$ por construção.

Considere que o algoritmo seja interrompido ao iterar sobre o ponto ik. Na primeira parte desta demonstração, será mostrado que sempre é possível reduzir qualquer problema ao caso especial em que todos os rótulos menores que ik foram usados, no momento da interrupção. O grafo G', nesse caso, consistirá de todos os pontos com rótulos menores que ik e todos os pontos além de ik que foram rotulados até o momento da interrupção. Na segunda parte, será mostrado em mais detalhes como obter G', com $\left\lceil \frac{|V(G')|-1}{diam(G')} \right\rceil > k$.

Suponha que o algoritmo seja interrompido na iteração ik e que alguns rótulos menores que esse valor não tenham sido usados na rotulação. Seja j o maior desses rótulos. Será mostrado que o algoritmo também teria sido interrompido no grafo G'' obtido de G ao se remover todos os vértices de rótulos menores ou iguais que j.

Se $j < (i-2)k$, então qualquer vértice com rótulo menor ou igual que j não tem qualquer efeito na rotulação dos cabelos do ponto ik, porque, para manter $bw_f(G) \leq k$ na rotulação dos cabelos do ponto ik, os rótulos usados devem ser maiores que $(i-2)k$. Então, pode-se remover os vértices com rótulos menores ou

iguais que j, para formar G''.

Se $(i-2)k < j < (i-1)k$, então, no início do algoritmo ser aplicado ao ponto ik, todos os $k-1$ rótulos entre $(i-1)k$ e ik estavam livres, porque qualquer rótulo entre $(i-1)k$ e ik só pode ser usado quando todos os rótulos entre $(i-2)k$ e $(i-1)k$ tiverem sido usados quando da rotulação dos pontos $(i-2)k$ e $(i-1)k$. Isso requer, no momento da interrupção, que o ponto ik não tenha cabelos de tamanho 2 não rotulados, e os rótulos menores ou iguais a j só servem para rotular vértices distantes 2 unidades de ik; os rótulos não usados estão muito distantes para serem usados na rotulação dos vértices a distância 1 do ponto ik. Então, pode-se remover de G os vértices com rótulos menores ou iguais que j, para formar G'', e o algoritmo 1 deve ser interrompido em G'' pelo mesmo motivo que seria interrompido em G.

O caso $(i-1)k < j$ é impossível porque o algoritmo não é interrompido até que todos os rótulos na faixa de $(i-1)k$ até $(i+1)k$ sejam usados.

Pode-se, portanto, considerar que, sempre que o algoritmo é interrompido, o grafo em análise tem todos os rótulos menores que ik em uso.

A partir de agora, será exibido detalhadamente como obter o subgrafo *caterpillar* G' tal que $\left\lceil \frac{|V(G')|-1}{diam(G')} \right\rceil > k$, em função da linha na qual o algoritmo foi interrompido.

1. O algoritmo é interrompido na linha 4.

   É impossível acontecer uma interrupção nesta linha porque ela não obriga a rotulação dos cabelos de tamanho 2, mas apenas indica rotulá-los em maior número possível.

2. O algoritmo é interrompido na linha 5.

   (a) Caso 1: nenhum cabelo foi rotulado na linha 4.

   O grafo G' consistirá de todos os pontos e os vértices com rótulos menores ou iguais que ik e todos os vértices distantes 1 unidade do ponto ik. O diâmetro de G' nesse caso é $i+1$ e o número de vértices é $(i+1)k+1$, referente aos rótulos de 0 até $(i+1)k$, mais um vértice, pelo menos, que não pode ser rotulado. Então, $|V(G')| \geq (i+1)k+2$, e $\left\lceil \frac{|V(G')|-1}{diam(G')} \right\rceil > k$.

   (b) Caso 2: pelo menos um cabelo foi rotulado na linha 4.

   O grafo G' consistirá do ponto ik e de todos os vértices distantes uma unidade desse ponto.

Como a linha 4 foi executada pelo menos uma vez, existia, no início da iteração, pelo menos um rótulo disponível entre $(i-2)k$ e $(i-1)k$ para rotular um cabelo de comprimento 2, logo nenhum dos rótulos maiores que $(i-1)k$ haviam sido usado. Como o algoritmo é interrompido na linha 5, todos os rótulos entre $(i-1)k$ e $(i+1)k$ foram dados aos vértices distantes 1 unidade do ponto ik, que compõem justamente o grafo G' construído neste caso. Além desses vértices, há pelo menos mais um que irá compor G', que é aquele que não pode ser rotulado e provocou a interrupção do algoritmo. Logo, $|V(G')| = 2k+2$ e $diam(G') = 2$, resultando $\left\lceil \frac{|V(G')|-1}{diam(G')} \right\rceil > k$.

3. O algoritmo é interrompido na linha 6.

   É impossível o algoritmo ser interrompido nesta linha pelos mesmos motivos dele não poder ser interrompido na linha 4.

4. O algoritmo é interrompido na linha 7.

   (a) Caso 1: nenhum cabelo foi rotulado na linha 6.

   Neste caso ocorrem os mesmos dois subcasos da interrupção na linha 5.

   (b) Caso 2: pelo menos um cabelo foi rotulado na linha 6.

   O grafo G' consistirá de todos os vértices (incluindo pontos) com rótulos menores ou iguais a ik e de todos os vértices distantes 1 ou 2 unidades do ponto ik. O grafo G' terá, então, os vértices de rótulos de 0 a $(i+1)k$, o ponto $(i+2)k$ (distante 2 unidades do ponto ik) e mais o vértice que não pode ser rotulado causando a interrução do algoritmo. E inclui-se também os vértices distantes 2 unidades do ponto ik que fazem parte dos cabelos de tamanho 2 que tiveram apenas os vértices de distância 1 rotulados pelo algoritmo, que totalizam $(i+1)k - 1 - (ik+1)$, e mais 1 referente ao do cabelo cujo vértice a distância 1 causou a interrupção do algoritmo. Logo, $|V(G')| = (i+2)k+3$ e $diam(G') = i+2$, resultando $\left\lceil \frac{|V(G')|-1}{diam(G')} \right\rceil > k$.

5. O algoritmo é interrompido na linha 8.

   O grafo G' consistirá dos vértices e dos pontos com rótulos menores ou iguais a ik, e de todos os vértices com distância 1 ou 2 do ponto ik.

Se a todos os vértices rotulados na linha 7 fossem dados rótulos maiores que ik (ou seja, entre $ik+1$ e $(i+1)k-1$), então aos vértices à distância 2, associados aos rotulados de distância 1, poderiam ser dados rótulos k unidades acima que seus correspondentes, e nesse caso o algoritmo não seria interrompido na linha 8. Então, pela contrapositiva da afirmação anterior, a algum vértice à distância 1 foi dado um rótulo menor ou igual a $ik-1$. O grafo G' incluirá todos os vértices (incluindo pontos) com rótulos entre 0 e $(i+1)k$, o ponto $(i+2)k$ e k vértices distantes 2 unidades de ik que não foram rotulados ainda. Assim, $|V(G')| = (i+2)k+2$ e $diam(G') = i+2$, logo $\left\lceil \frac{|V(G')|-1}{diam(G')} \right\rceil > k$.

□

Teorema 4.2. *O bandwidth de um grafo caterpillar G com cabelos de tamanho no máximo 2 é o maior valor dentre todos os valores de $\left\lceil \frac{|V(G')|}{diam(G')} \right\rceil$ para todos os subgrafos caterpillar G' de G.*

Demonstração. Aplicar o algoritmo 1 com

$$k = \max \left\{ \left\lceil \frac{|V(G')|}{diam(G')} \right\rceil \ : G' \text{ subgrafo } \textit{caterpillar} \text{ de } G \right\}$$

Se o algoritmo não for interrompido, então $bw(G) \leq k$. O resultado do teorema é consequência direta do lema 2.2. □

Corolário 4.1. *Existe um algoritmo $O(n \log n)$ para encontrar o bandwidth de um grafo caterpillar G com cabelos de tamanho no máximo 2, que também produz uma rotulação desse valor de $bw(G)$.*

Demonstração. O algoritmo 1 tem complexidade $O(n)$ porque percorre todos os pontos e vértices de G durante a rotulação do grafo. Então, para se encontrar o valor de $bw(G)$, executa-se uma busca binária sobre a faixa de possíveis valores de $bw(G)$: de 1 a $n-1$. A complexidade resultando é $O(n \log n)$. Durante a execução do algoritmo, é feita a rotulação do grafo. □

4.2 Intervalo

Em 1987, Kratsch [23] apresentou o que seria o primeiro algoritmo polinomial $O(n^2 \log n)$ para BANDWIDTH em grafos de intervalo, mas ele continha um erro. Três anos depois, em 1990, Kleitman e Vohra [19] apresentaram um algoritmo polinomial $O(nk)$ correto, k BAND, para BANDWIDTH em grafos de intervalo, que é considerado o primeiro algoritmo polinomial para essa classe de grafos.

Em 1991 e em 1994, dois artigos — respectivamente, um de Mahesh, Rangan e Srinivasan [27], e o outro de Sprague [33] — foram publicados exibindo o erro de [23]. Entretanto, um não faz referência ao outro, o que leva a crer que ambos foram observações independentes. O artigo de Sprague apenas mostra o erro de Kratsch, mas não apresenta uma solução; já o artigo de Mahesh, Rangan e Srinivasan apresenta o erro de Kratsch além de adaptar o algoritmo original para resolver corretamente o problema, mantendo a complexidade.

Sprague — no mesmo artigo [33] — desenvolveu um novo algoritmo polinomial $O(n \log n)$ para BANDWIDTH, baseado no algoritmo de Kleitman e Vohra. Nessa nova versão, foram feitas mudanças apenas nas estruturas de dados usadas; o algoritmo em si permaneceu o mesmo. Esta nova abordagem traz um melhor (menor) limite superior para o problema quando se considera valores grandes de k comparados a $\log n$.

Esta seção apresenta a solução polinomial do problema BANDWIDTH para grafos de intervalo de Kleitman e Vohra, concentrando-se em descrever o funcionamento do algoritmo e em refazer a sua demonstração de complexidade.

Definição (Grafo de interseção)**.** *Um grafo $G = (V, E)$ é de interseção de uma família de conjuntos $\mathcal{F}$, denotado por $G = \Omega(\mathcal{F})$, se existe uma função bijetora que mapeia os vértices $v \in V$ em conjuntos $F_v \in \mathcal{F}$, tal que $uv \in E$ se, e somente se, $F_u \cap F_v \neq \emptyset$.*

Definição (Grafo de intervalo)**.** *Um grafo $G = (V, E)$ é de intervalo se é um grafo de interseção de uma família $\mathcal{I}$ de intervalos na reta real, onde cada intervalo $I_v \in \mathcal{I}$ pode ser visto como um intervalo $[a_v, b_v]$, em que $a_v, b_v \in \mathbb{R}$ em uma reta real.*

A solução do problema é dada pelo algoritmo 2, cuja demonstração de corretude mostra que a rotulação construída sempre irá satisfazer $bw \leq k$ se $G = \Omega(\mathcal{I})$, dado na entrada, tiver $bw \leq k$; e, caso contrário, com $bw > k$, a execução será

Algoritmo 2 k BAND: calcular o *bandwidth* de um grafo de intervalo, em tempo $O(nk)$.

Entrada: $k \in \mathbb{N}$ e família $\mathcal{I}$ de intervalos na reta real na qual $G = \Omega(\mathcal{I})$, com $n = |\mathcal{I}|$.
Saída: **verdadeiro** $\Leftrightarrow bw(G) \leq k$

para $I_i \in \mathcal{I}$ **faça**
  $f(I_i) \leftarrow 0$ e $M(I_i) \leftarrow n$
fim para
$q \leftarrow 0$ e $U \leftarrow \mathcal{I}$
Selecionar $I_i \in U$ com menor b_i
enquanto $U \neq \emptyset$ **faça**
  $q \leftarrow q + 1$
  $f(I_i) \leftarrow q$ e $U \leftarrow U \setminus \{I_i\}$
  para $I_r \in U : I_r \cap I_i \neq \emptyset$ e $M(I_r) = n$ **faça**
    $M(I_r) \leftarrow \min\{f(I_i) + k, n\}$
  fim para
  $S_j^q \leftarrow \{I_r \in U : M(I_r) \leq q + j\}$
  se $\exists j : |S_j^q| > j$ **então**
    retorne **falso**
  senão $\{\forall j : |S_j^q| \leq j\}$
    Encontrar o menor j_0 tal que $|S_{j_0}^q| = j_0$
    Selecionar $I_i \in S_{j_0}^q$ com menor b_i
  fim se
fim enquanto
retorne **verdadeiro**: todos os intervalos já foram rotulados

interrompida antes de todos os intervalos serem rotulados. A tabela 4.1 contém as variáveis usadas no algoritmo 2, k BAND.

Variável	Descrição
I_i	intervalo I_i
a_i	extremo esquerdo do intervalo I_i
b_i	extremo direito do intervalo I_i
$f(I_i)$	rótulo do intervalo I_i, que possui valor inicial 0
$M(I_i)$	marca do intervalo, que corresponde a um limite superior para o valor de rótulo; o valor inicial é n
q	contador de iterações
U	conjunto de intervalos não rotulados; quando um intervalo recebe um rótulo, ele é removido deste conjunto

Tabela 4.1: Variáveis usadas no algoritmo k BAND.

A ideia do algoritmo é, a cada iteração, rotular um intervalo I_i, e agendar todos os outros intervalos adjacentes a I_i ainda não agendados para serem rotulados até uma iteração k unidades no futuro. Se um intervalo I_p agendado para ser rotulado até uma iteração q' qualquer não for rotulado até lá, então qualquer rótulo dado a I_p depois de q' iria provocar uma diferença de rótulos maior que k entre I_p e o intervalo adjacente a I_p que o agendou, fazendo com que se saiba *a priori* que $bw > k$, e o algoritmo saberia que poderia parar neste momento. Para escolher o próximo intervalo a ser rotulado, portanto, o algoritmo escolhe, a cada iteração q, o conjunto de intervalos S_j^q tal que $|S_j^q| = j$, ou seja, o conjunto de intervalos agendados que esteja numa situação crítica, de forma que a não escolha de um intervalo desse conjunto irá provocar uma rotulação com $bw > k$.

Da ideia anterior, fica a pergunta se sempre vai existir um conjunto S_j^q com $|S_j^q| = j$ para que possa escolher um vértice. E a resposta a esse questionamento é sim, porque o último conjunto S_{n-q}^q tem todos os vértices ainda não rotulados e é sempre verdade que $|S_{n-q}^q| = n - q$.

Nos próximos parágrafos seguem alguns detalhes quanto à manipulação das variáveis do algoritmo. Usa-se marca de um vértice para denotar o que foi tratado anteriormente como agendamento.

Seja $\mathcal{I}$ a família de intervalos dada na entrada do algoritmo. Na iteração q, cada conjunto S_j^q contém os intervalos não rotulados I_r de $\mathcal{I}$ que possuem marca $M(I_r) \leq q + j$. Como $i \leq j$ implica $S_i^q \subseteq S_j^q$, nota-se que os conjuntos S_j^q não precisam ser totalmente reconstruídos, mas apenas atualizados. Tal facilidade permite que o

algoritmo tenha complexidade $O(n^2 \log n)$. A atualização dos conjuntos S_j^q é feita pelas seguintes regras.

$$S_j^{q+1} = S_{j+1}^q \setminus \{I_i\}, \quad j \leq k-1 \tag{4.1}$$

$$S_k^{q+1} = S_{k-1}^{q+1} \cup \{I_r \in U : \ I_r \cap I_i \text{ e } M(I_r) = n\} \tag{4.2}$$

$$S_j^{q+1} = S_k^{q+1}, \quad k+1 \leq j \leq (n-1)-(q+1) \tag{4.3}$$

O conjunto $S_{n-(q+1)}^{q+1}$ contém os intervalos ainda não rotulados.

Sobre quais conjuntos S_j^q existem em cada iteração. Analisando a primeira iteração do laço **enquanto**, percebe-se que o rótulo atribuído a um intervalo I_i é $f(I_i) = q = 1$ e, consequentemente, o menor dos valores $M(I_r)$ atribuídos na linha 10 é $\min\{f(I_i)+k, n\} = \min\{1+k, n\}$. O maior valor j entre os conjuntos S_j^q é $n-1$, já que $q \geq 1$ e $M(I_r) \leq n$ para todo I_r. Considerando $k \leq n-1$, os conjuntos S variam inicialmente de S_k^q até S_{n-1}^q. A medida que o valor de q aumenta, o valor mínimo de j em S_j^q naturalmente diminui. A figura 4.2 ilustra uma possível evolução dos conjuntos para o caso $n = 7$ e $k = 3$.

Iteração (q)			S_k^q			
1			S_3^1	S_4^1	S_5^1	S_6^1
2		S_2^2	S_3^2	S_4^2	S_5^2	
3	S_1^3	S_2^3	S_3^3	S_4^3		
4	S_1^4	S_2^4	S_3^4			
5	S_1^5	S_2^5				
6	S_1^6					

Figura 4.4: Exemplo da evolução dos conjuntos S_j^q para $n = 7$ e $k = 3$.

Na equação (4.1), remover o intervalo I_i do conjunto S_{j+1}^q na atualização não implica obrigatoriamente reduzir o tamanho do conjunto. Se $I_i \notin S_{j+1}^q$, então $S_j^{q+1} = S_{j+1}^q$. Essa particularidade pode ocorrer porque I_i escolhido na linha 17 da iteração q depende da escolha de j_0 na linha 16, e que j_0 pode não ser menor ou igual a $k-1$ (resta a opção de ser igual a k). A nomenclatura nesse caso pode induzir ao erro, embora não esteja matematicamente incorreta.

A construção dos conjunto S_j^q na linha 12 sempre coloca novos intervalos — que pertencem a U e são adjacentes ao recém rotulado — no conjunto S_k^q. Pela regra

de atualização 4.1, isso implica que o conjunto S_j^q escolhido na linha 16 mantém a característica $q + j = q_0 + k$, sendo q_0 a iteração inicial; essa propriedade de atualização obriga o algoritmo a rotular intervalos escolhidos de um conjunto $S_{j_0}^q$ no passos 17 com $j_0+q \geq q_0+k$. Supondo o caso crítico $j_0+q = q_0+k$ com $j_0 \geq 1$, tem-se que $q - q_0 = k - j_0 \leq k$. Então a diferença de rótulos entre intervalos adjacentes respeita a característica $bw \leq k$, enquanto o processo de rotulação continuar no algoritmo, até o momento em que não há mais intervalos a serem rotulados e o algoritmo retorna **verdadeiro** na linha 20.

Se o algoritmo retorna **falso** na linha 14, então existe um j' — considera-se o primeiro deles — em que $|S_{j'}^q| > j'$. Será mostrado no lema 4.1 que isso implica $|S_k^q| > k$, que significa que existem pelo menos $k + 1$ intervalos não rotulados após um intervalo ter sido rotulado. Assim, um dos intervalos em S_k^q deve receber um rótulo de no mínimo $q+k+1$ e como cada $I_i \in S_j^q$ possui um intervalo adjacente com rótulo $\leq k$, então não é possível k BAND produzir uma rotulação com $bw \leq k$. Este argumento não é suficiente para provar que o algoritmo retorna falso se e somente se $bw(G) > k$; o lema 4.2 complementa o restante da demonstração.

Lema 4.1. *Se o algoritmo 2 retorna falso, então para algum k existe um j' tal que $|S_{j'}^q| > j'$, e isso implica que $|S_k^q| > k$.*

Demonstração. Suponha por contradição que $|S_k^q| \leq k$ para o caso em questão. No passo anterior, $q - 1$, o algoritmo não foi interrompido, logo $|S_l^q| \leq l$ para todo l.

Como $|S_{j+1}^{q-1}| \leq j + 1$ para todo $j + 1 \leq k$, considera-se os seguintes casos:

- $|S_{j+1}^{q-1}| < j + 1$

  Pela regra 2, $|S_j^q| \leq |S_{j+1}^{q-1}|$. $|S_{j+1}^{q-1}| < j + 1 \Leftrightarrow |S_{j+1}^{q-1}| \leq j$. $|S_j^q| \leq |S_{j+1}^{q-1}| \leq j \Leftrightarrow |S_j^q| \leq j$

- $|S_{j+1}^{q-1}| = j + 1$

  - Se j+1 for o primeiro inteiro para o qual a igualdade ocorre, então a remoção do intervalo I_i, que foi rotulado em q, fará com que a regra de atualização $S_j^q = S_{j+1}^{q-1} \setminus \{i\}$ remova de fato I_i do conjunto anterior, logo $|S_j^q| = |S_{j+1}^{q-1}| - 1 = (j + 1) - 1 = j \Rightarrow |S_j^q| \leq j$.
  - Se j+1 não for o primeiro inteiro para o qual a igualdade ocorre, a regra anterior permanece válida, porque o elemento rotulado é excluído do

conjunto U e, pela regra 1, será excluído de todos os conjuntos S^q_{j+1} à direita.

Como $S^q_j = S^q_k$ para todo $k \leq j < n-q$, então (usando a suposição que $|S^q_k| \leq k$) $|S^q_j| = |S^q_k| \leq k \leq j$ para $k \leq j < n-q$. Adicionalmente, $|S^q_{n-q}| = n-q$. Assim, $|S^k_j| \leq j$ para todo j: contradição, logo $|S^q_k| > k$ se k BAND termina na linha 14. □

Lema 4.2. *Quando o algoritmo 2 retorna falso para uma família de intervalos $\mathcal{I}$ e um inteiro k na iteração q, então todo $I_\alpha \in S^q_k$ é adjacente ao intervalo I_β tal que $f(I_\beta) = q$.*

Demonstração. Suponha que exista I_α a esquerda de I_β ($b_\alpha < a_\beta$) em S^q_k. Como $a_\alpha \leq a_\beta$, então $M(I_\alpha) \leq M(I_\beta)$. Assim, para todo S^q_j ao qual I_β pertença, I_α também pertence. Isso implica que, no momento de escolher I_β para ser rotulado $f(I_\beta) = q$, I_α também pertencia ao conjunto S^{q-1}_j, logo I_α seria escolhido pelo algoritmo para ser rotulado $f(I_\alpha) = q$: contradição.

Suponha que exista I_α a direita de I_β em S^q_k. Como I_α não é adjacente a I_β mas é marcado — pois pertence a S^q_k —, então algum intervalo que termina depois de I_β foi rotulado antes de I_β. Seja I_δ o intervalo escolhido que foi responsável por marcar I_α. Se I_β inicia primeiro que I_δ, a marca de I_β seria menor que a de I_δ, e I_β estaria disponível em todos os S^p_j em que I_δ estaria disponível. Assim, I_β seria rotulado antes de I_δ. Como isso não ocorreu, então I_δ inicia primeiro que I_β. Logo, o intervalo I_β está propriamente contido no intervalo I_δ. Assim, na iteração q, nenhum intervalo I_r seria marcado com $M(I_r) = q + k$, pois todos os adjacentes a I_β teriam sido marcados quando I_δ foi rotulado. Como $|S^q_k| > k$, então $|S^{q-1}_k| > k$, e o algoritmo teria parado na iteração anterior. □

Lema 4.3. *Sejam I_α e I_β dois intervalos tais que nenhum deles têm o menor b_i dentre os intervalos de $\mathcal{I}$. Se $a_\alpha \leq a_\beta$, então o algoritmo 2 obtém que $M(I_\alpha) \leq M(I_\beta)$.*

Demonstração. Essa é uma consequência direta do algoritmo 2. Se por contradição $M(I_\alpha) > M(I_\beta)$, então existiria um caminho entre o intervalo de menor b_i e o intervalo I_β sem que nenhum intervalo intermediário fosse adjacente a I_α, o que impossível pois o grafo é de intervalo. □

Teorema 4.3. *O algoritmo 2 determina se o bandwidth de um grafo de intervalo é menor ou igual a k em $O(nk)$ passos.*

Demonstração. Já foi visto que, se o algoritmo termina na linha 20, a família de intervalos $\mathcal{I}$ de entrada, na qual $G = \Omega(\mathcal{I})$ possui uma rotulação f onde $bw(G) \leq bw_f(G) \leq k$, por construção. Resta mostrar então que, se o algoritmo termina na linha 14, então $bw(G) > k$, ou seja, não existe uma rotulação f' de $\mathcal{I}$ que admita $bw_{f'}(G) \leq k$.

Suponha por contradição que o algoritmo pare na linha 14, mas exista uma rotulação f' com $bw_{f'}(G) \leq k$. Seja G o menor (n mínimo) grafo de intervalo para o qual tal fato ocorre. Será mostrado que é possível construir um grafo de intervalo G' menor que G tal que o algoritmo também pare na linha 14 mas exista f'' de $\mathcal{I}'$ que admite $bw_{f''}(G') \leq k$: contradição, porque G é o menor.

Supondo que o algoritmo termine na linha 14, $|S_k^q| \geq k+1$ (lema 4.1). Seja I_β o último intervalo rotulado: $f(I_\beta) = q$. Para todo $I_\alpha \in S_k^q$, $M(I_\alpha) \leq q + k$ e, pelo lema 4.2, $a_\alpha \leq b_\beta$. Se $b_\alpha \geq b_\beta$ para todo $I_\alpha \in S_k^q$, então o conjunto $S_k^q \cup \{I_\beta\}$ forma uma clique de tamanho maior ou igual a $k+2$. Neste caso, $\mathcal{I}$ não pode admitir rotulação f' tal que $bw_{f'}(G) \leq k$, logo deve existir $I_\gamma \in S_k^q$ tal que $b_\gamma < a_\alpha \leq b_\beta$ para algum I_α, conforme ilustração na figura 4.5.

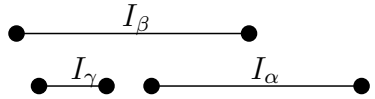

Figura 4.5: Posição dos intervalos I_α, I_β e I_γ para não haver clique de tamanho $\geq k+2$.

Embora $b_\gamma < b_\beta$, γ não foi selecionado para ser rotulado na iteração anterior (já que I_γ ainda não está rotulado na iteração atual). Assim, I_γ não pertencia ao conjunto S_j^{q-1} para o menor dos inteiros j tal que $|S_j^{q-1}| = j$, $j \leq k$. Em particular, se $j = k$ em $q-1$, então $I_\gamma \in S_{n-q+1}^{q-1}$ na iteração $q-1$ e $I_\gamma \in S_k^q$ na iteração q, *i.e.*, I_γ foi marcado por I_β.

De forma geral, todos os intervalos em S_j^{q-1} interceptam ou o intervalo I_δ de rótulo $f(I_\delta) = j + q - 1 - k$ (caso dos intervalos I_r com $M(I_r) = j + (q-1)$) ou um intervalo I_ϵ com $b_\epsilon < b_\delta$ (caso dos intervalos I_r com $M(I_r) < j + (q-1)$); essa propriedade pode ser visualizada com o uso do lema 4.3. Como, na iteração $q-1$, $b_\beta \leq b_\theta$ para todo $I_\theta \in S_j^{q-1}$ (já que I_β foi o escolhido), então todo intervalo de S_j^{q-1} intercepta ou o intervalo I_δ ou, por consequência óbvia, todos os intervalos I_λ tal que $b_\lambda \geq b_\gamma$.

Na figura 4.6, pode-se visualizar a organização dos intervalos I_δ, I_β, I_γ e I_λ.

Pelo lema 4.3, se $a_\gamma \leq a_\beta$, $M(I_\gamma) \leq M(I_\beta)$ e I_γ seria selecionado no lugar de I_β na iteração $q-1$, logo $a_\beta < a_\gamma$. Como $I_\gamma \notin S_j^{q-1}$, então $b_\delta < a_\gamma$; caso contrário, I_γ teria sido rotulado por I_δ e pertenceria a S_j^{q-1}. Por esta organização dos intervalos, fica fácil perceber que todos os intervalos $I_\theta \in S_j^{q-1}$ têm $b_\beta < b_\theta$, pois I_β foi selecionado na iteração $q-1$; e $a_\theta \leq b_\delta$, pois $I_\theta \in S_j^{q-1} \Rightarrow a_\theta \leq b_\delta$.

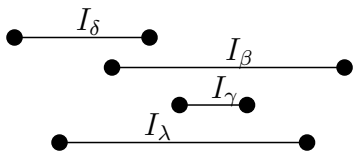

Figura 4.6: Exemplificação da organização dos intervalos I_δ, I_β, I_γ e I_λ.

Seja $G' = \Omega(\mathcal{I}')$ o grafo de interseção da família $\mathcal{I}'$ de intervalos na reta real obtida pela remoção dos intervalos $S_k^q \setminus S_j^{q-1} - \{I_\gamma\}$ de $\mathcal{I}$ (ver figura 4.7) bem como todos os intervalos com marca excedendo $q+k$ (aqueles ainda não marcados). Além disso, estenda I_γ à esquerda até que fique adjacente a I_δ. Considere a rotulação f'' de $\mathcal{I}'$ como a numeração consecutiva dos intervalos de $\mathcal{I}'$ na mesma ordem da rotulação f' de $\mathcal{I}$.

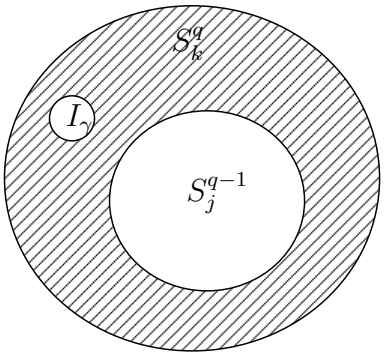

Figura 4.7: Diagrama exibindo os intervalos que devem ser removidos de $\mathcal{I}$ — marcados com hachura.

Primeiro, como consequência da remoção dos intervalos, o grafo G' construído é menor que G. Pois $I_\beta \in S_j^{q-1}$ e $I_\gamma \notin S_j^{q-1}$ implicam $S_j^{q-1} \subsetneq S_k^q$ e $j \leq k-1$. E $|S_k^q| \geq k+1$ e $S_j^{q-1} = j \leq k-1$ implicam, pelo menos, $|S_k^q \setminus S_j^{q-1}| - 1 \geq 2 - 1 = 1$ vértice removido de $\mathcal{I}$.

Segundo, o *bandwidth* de G' não pode ter sido aumentado pela remoção dos vértices. E a extensão do intervalo I_γ, à esquerda, até interceptar o intervalo I_δ não

criou diferença de rótulos maior do que k, porque se foi criada uma nova adjacência entre I_γ e um intervalo qualquer I_ρ, então $b_\rho < a_\gamma$ (já que foi criada uma nova adjacência) e há duas possibilidades: (1) $a_\rho \leq b_\delta$ e todos os intervalos aos quais serão adjacentes a I_ρ terão rótulos entre o rótulo de I_δ e o rótulo de I_γ, cujas diferenças permanecem menores ou iguais a k; (2) $a_\rho > b_\delta$ implica I_ρ ter sido excluído na remoção dos vértices.

Por último, sabe-se que $I_\beta \in S_j^{q-1}$, logo $M(I_\beta) \leq j+q-1$. E também sabe-se que os últimos intervalos marcados em $\mathcal{I}'$ foram marcados ou antes de I_β ou junto com I_β, inclusive I_γ pela sua extensão à esquerda; todos têm, portanto, marcas menores ou iguais a $j+q-1$. Como existem $q+j$ intervalos em $\mathcal{I}'$ — q rotulados até o momento da escolha de I_γ mais j pelo tamanho de S_k^q de G' —, o algoritmo aplicado a $\mathcal{I}'$ para na linha 14, ou seja, não é possível rotular $q+j$ intervalos com $q+j-1$ rótulos tal que $bw(G') \leq k$. Existe, dessa forma, um grafo $G' = \Omega(\mathcal{I}')$ menor que $G = \Omega(\mathcal{I})$ cuja rotulação f'' implica $bw_{f''}(G') \leq k$. □

4.2.1 Intervalo unitário

Definição. *Definição (Grafo de intervalo unitário). Um grafo $G = (V, E)$ é de intervalo unitário se é o grafo de interseção de uma família $\mathcal{I}$ de intervalos de mesmo tamanho na reta real.*

Para a subclasse de grafos de intervalos unitários, que é equivalente a classe de grafos de intervalo próprio [11], o teorema 4.4 apresenta a rotulação ótima para o BANDWIDTH de um grafo G.

Teorema 4.4 (*bandwidth* de grafos de intervalo unitário)**.** *Seja $G = \Omega(\mathcal{I})$ um grafo de intervalo unitário, uma rotulação ótima de G para o problema bandwidth é dada pela atribuição dos menores rótulos possíveis aos intervalos de $\mathcal{I}$ na ordem crescente dos extremos esquerdos.*

Demonstração. Seja f a rotulação dos intervalos tal que os rótulos de 1 a n sejam dados a eles na ordem crescente dos seus extremos esquerdos, e sejam I_i e I_j dois intervalos com $I_i \cap I_j \neq \emptyset$ (vide figura 4.9) cuja diferença $|f(I_i) - f(I_j)|$ seja a maior existente: $bw_f(G) = |f(I_i) - f(I_j)|$. Em função da rotulação e do tamanho fixo dos intervalos, há $|f(I_j) - f(I_i)| - 1$ intervalos entre o início de I_i e o início de I_j, conferindo a existência de uma clique de tamanho $|f(I_j) - f(I_i)| + 1$ (incluindo I_i e

I_j). Dado que $bw(G) \geq \omega(G) - 1 \geq |f(I_j) - f(I_i)| + 1 - 1$ (usando a equação (2.10)) e que $bw_f(G) = |f(I_i) - f(I_j)|$, conclui-se que f é uma rotulação ótima.

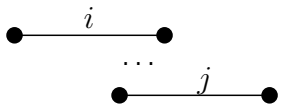

Figura 4.8: Intervalos i e j adjacentes com maior diferença nas rotulações $f(i)$ e $f(j)$.

□

Corolário 4.2. *O bandwidth de um grafo de intervalo unitário G é igual a $\omega(G) - 1$.*

4.3 *Quasi-threshold*

A classe dos grafos *quasi-threshold* possui um algoritmo linear para resolver BANDWIDTH [37].

Em 1995, Yan, Chen e Chang [37] apresentaram um resultado sobre como rotular os vértices de um grafo *quasi-threshold* G de maneira ótima, permitindo calcular o valor de $bw(G)$ diretamente, aplicando esse resultado na árvore que caracteriza cada grafo dessa classe.

Dois anos depois, em 1997, Yan [35], sozinho, apresenta um resultado mais abrangente para a classe de cografos, que inclui *quasi-threshold*. Esse outro resultado, análogo ao anterior, permite calcular diretamente o *bandwidth* de um grafo G ao aplicá-lo na coárvore de G.

E mais um ano depois, em 1998, novamente, Yan [36] estende o resultado de cografos para a classe de P_4-esparsos, permitindo, da mesma forma que na classes mais restritas, calcular diretamente o *bandwidth* de um grafo G ao aplicar diretamente esse resultado na árvore caracterizadora de G.

Nas próximas seções, serão apresentados os três resultados encontrados para *quasi-threshold*, cografos e P_4-esparsos.

Definição (*Quasi-threshold*)**.** *A classe de grafos quasi-threshold é definida recursivamente como segue.*

1. *um grafo trivial é quasi-threshold;*
2. *a junção de um grafo quasi-threshold com um grafo trivial é quasi-threshold;*
3. *a união de dois grafos quasi-threshold é quasi-threshold.*

Wolk [34] chama os grafos desta classe de grafos de comparabilidade de árvore, por ser possível uma caracterização deles pelo fecho transitivo de uma árvore enraizada direcionada (mais detalhes abaixo). Golumbic [10] chama-os de trivialmente perfeitos com relação a um conceito de "perfeição". E Ma *et al* [26] chamam-os de *quasi-threshold*. Será usado nesta dissertação o nome dado por Ma *et al* [26].

O artigo de Ma *et al* apresenta um algoritmo de reconhecimento $O(mn)$ para a classe e um algoritmo $O(n^2)$ para resolver BANDWIDTH.

Quasi-threshold é uma subclasse de cografo, e é subclasse também de intervalo. O algoritmo linear para BANDWIDTH (Ma *et al* [26]) é similar ao algoritmo linear

do mesmo problema para cografos [35]. Ambos são baseados no percurso de árvores caracterizadoras. Para cografos, há a coárvore [3]; e para *quasi-threshold* há a árvore enraizada direcionada, que foi apresentada originalmente em Yan *et al* [37].

Uma **árvore enraizada** R_T é um grafo direcionado obtido de uma árvore T ao atribuir uma direção para cada aresta de T, de tal forma que exista um vértice r, chamado raiz, a partir do qual possua um caminho direcionado para cada vértice de $T \setminus \{r\}$. O grafo $G(R_T) = (V, E')$ induzido por $R_T = (V, E)$ possui $uv \in E'$ se, e somente se, existe um caminho direcionado de u até v.

Teorema 4.5 (*Quasi-threshold*)**.** *Um grafo G é quasi-threshold se, e somente se, G é induzido por uma árvore enraizada.*

Demonstração. Seja G um grafo *quasi-threshold* caracterizado pela operações de união e junção, de acordo com a definição

A operação de união é representada em árvores direcionadas enraizadas pela união disjunta das árvores, e a operação de junção de um grafo trivial v com um grafo *quasi-threshold* G, pela inclusão de v como nova raiz da árvore $T + v$ representativa de $G + v$. A inclusão de v como nova raiz de $T + v$ garante que v alcance todos os vértices de T, uma vez que a raiz de T já tinha essa propriedade, e v agora, por construção, alcança a raiz de T.

E se T é uma árvore enraizada, pode-se construir G pela operação de remover a raiz v de T, considerando v um vértice universal de G. Se a árvore $T \setminus \{v\}$ for conexa, repete-se a operação de remover a raiz; caso não, então cada sub-árvore representa recursivamente um grafo *quasi-threshold*. □

Um exemplo de um grafo *quasi-threshold* e sua árvore enraizada é apresentado na figura 4.9.

O algoritmo 3 verifica se um grafo de entrada G é *quasi-threshold*, e retorna a árvore enraizada de G em caso afirmativo ou o conjunto vazio caso contrário. A notação $d^-(u)$ no algoritmo significa o grau entrada do vértice u, e $d(u)$, o grau do vértice u, conforme notação na seção 1.1.

O teorema 4.6 é fundamental para a construção do algoritmo para grafos *quasi-threshold*. O uso deste resultado, bem como das equações e inequações (2.5), (2.6) e (2.8), compõem as operações necessárias para o algoritmo 4, que calcula o *bandwidth* para um grafo *quasi-threshold*.

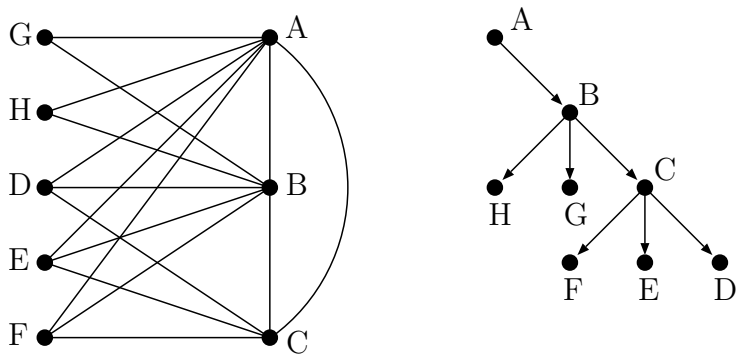

Figura 4.9: Exemplo de um grafo *quasi-threshold* e sua árvore enraizada.

Algoritmo 3 QT: verificar se um grafo é *quasi-threshold*, em tempo $O(n)$.

Entrada: Um grafo $G = (V, E)$
Saída: Um árvore enraizada R de G se, e somente se, G é *quasi-threshold*

1: calcular $d(v_i)$ para cada vértice v_i de G
2: **para** $i = 0$ to n **faça**
3: $d^-(v_i) \leftarrow 0$
4: **fim para**
5: **para** cada $v_iv_j \in E$ **faça**
6: **se** $d(v_i) > d(v_j)$ or $(d(v_i) = d(v_j)$ and $i < j)$ **então**
7: $d^-(v_j) \leftarrow d^-(v_j) + 1$
8: **senão**
9: $d^-(v_i) \leftarrow d^-(v_i) + 1$
10: **fim se**
11: **fim para**
12: $F \leftarrow \emptyset$
13: **para** $j = 1$ to n **faça**
14: **se** $d^-(j) \geq 1$ **então**
15: escolher um vértice $v_i \in N[v_j]$ tal que $d(v_i) > d(v_j)$ ou $(d(v_i) = d(v_j)$ e $i < j)$ e $d^-(i)$ é o maior;
16: **fim se**
17: adicionar o arco iv_j em F
18: **fim para**
19: usar uma busca em profundidade para calcular o número $anc(v_j)$ de antecessores de v_j em F para cada j
20: **se** $\forall j, d^-(v_j) = anc(v_j)$ **então**
21: **retorne** F
22: **senão**
23: **retorne** $\emptyset$
24: **fim se**

Teorema 4.6. *Considere* $G_1 = (V_1, E_1), \ldots, G_r = (V_r, E_r)$ *grafos disjuntos com* $n_1 \geq \ldots \geq n_r$ *e* $m_i = \sum_{j=1}^{i} n_j$ *para* $1 \leq i \leq r$*, sendo* $n_i = |V_i|$*. O conjunto dos vértices de cada grafo* G_i *é dado por* $V_i = \{v_{m_{i-1}+1}, \ldots, v_{m_i}\}$ *para* $1 \leq i \leq r$*. Seja* f *uma rotulação de* G_1 *com* $f(v_j) = j$ *para* $1 \leq j \leq n_1$*, e seja* $G = (V, E)$ *o grafo obtido por* $\cup_{1 \leq i \leq r} G_i$ *adicionado com um vértice universal* v_0*, como pode ser visto na figura 4.10. Considere a seguinte rotulação* g *de* G *como segue.*

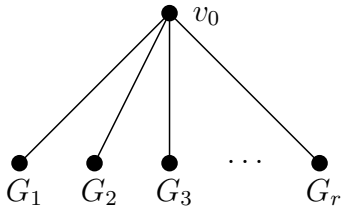

Figura 4.10: Grafo G obtido pela junção de um vértice universal v_0 a r grafos disjuntos $G_1, \ldots, G_r$.

$$g(v_j) = \begin{cases} j, & se\ 1 \leq j \leq \left\lceil \frac{m_r}{2} \right\rceil \\ \left\lceil \frac{m_r}{2} \right\rceil + 1, & se\ j = 0 \\ j + 1, & se\ \left\lceil \frac{m_r}{2} \right\rceil + 1 \leq j \leq m_r \end{cases}$$

Se $n_1 \leq \left\lceil \frac{m_r}{2} \right\rceil$*, então* g *é uma rotulação ótima de* G*. Se* $n_1 \geq \left\lceil \frac{m_r}{2} \right\rceil + 1$ *e* f *for uma rotulação ótima, então* g *é uma rotulação ótima de* G*.*

O valor de $bw(G)$ *é* $\max\left\{\left\lceil \frac{n}{2} \right\rceil, bw(G_1) + 1\right\}$*, usando a rotulação definida anteriormente.*

Demonstração. Com a rotulação definida anteriormente, $bw(G) \leq \max\left\{\left\lceil \frac{n}{2} \right\rceil, bw(G_1) + 1\right\}$ porque o algoritmo atinge esse valor ao incluir o vértice v_0 no centro da rotulação.

Se $bw(G_1) \leq \left\lceil \frac{n}{2} \right\rceil - 1$, então $bw(G) = \left\lceil \frac{n}{2} \right\rceil$ é ótimo pelo limite inferior (2.5).

Se $bw(G_1) \geq \left\lceil \frac{n}{2} \right\rceil$, temos que provar que $bw(G_1)+1$ é um valor ótimo para $bw(G)$. Suponha, então, por contradição que o algoritmo retorna $bw(G) = bw(G_1) + 1$ quando, na verdade, $bw(G) = bw(G_1)$. Como $bw(G) = bw(G_1) \geq \left\lceil \frac{n}{2} \right\rceil$, qualquer que seja a rotulação ótima usada, vai existir uma aresta uv em G que passa pelo rótulo $\left\lceil \frac{n}{2} \right\rceil + 1$ (podendo, inclusive, o vértice com esse rótulo ser ou u ou v). Nesse caso, se o vértice v_0 incluído tiver rótulo entre o rótulo de u e o de v, então $bw(G) = bw(G_1) + 1$. Por outro lado, se v_0 tiver rótulo menor que u ou maior que v, então, por v_0 ser universal, deve existir uma aresta de v_0 para o rótulo 1 e uma aresta de

v_0 para o rótulo n, cujas diferenças são maiores que $bw(G_1)$, e $bw(G) \geq bw(G_1) + 1$: contradição. □

O algoritmo 4, descrito a seguir, encontra o valor de *bandwidth* de um grafo *quasi-threshold*, em tempo $O(n)$.

Algoritmo 4 Computar $bw(G)$ de um grafo G *quasi-threshold*, em tempo $O(n)$.

Entrada: Um grafo $G = (V, E)$
Saída: $bw(G)$

1: Verificar se G é *quasi-threshold* com o algoritmo 3; terminar caso o grafo não seja desta classe
2: Obter a árvore enraizada T_G de G pelo passo anterior
3: Rotular os vértices de T_G em $1, \ldots, |V(T_G)|$ por um percurso em pós-ordem
4: **para** $i = 1, \ldots, |V(T_G)|$ **faça**
5: $\quad m(v_i) \leftarrow 1$
6: $\quad r(v_i) \leftarrow 1$
7: $\quad bw(v_i) \leftarrow 0$
8: **fim para**
9: **para** $i = 1, \ldots, n$ **faça**
10: $\quad$ **se** v_i não é folha **então**
11: $\quad\quad$ Sejam $u_1, \ldots, u_k$ os filhos de v_i
12: $\quad\quad m(v_i) \leftarrow \sum_{j=1}^{k} m(u_j) + 1$
13: $\quad\quad r(v_i) \leftarrow \left\lceil \frac{m_i}{2} \right\rceil + 1$
14: $\quad\quad$ Rerotular os vértices de G_{v_i} de acordo com o teorema 4.6
15: $\quad\quad u_{max} \leftarrow \{u_j \ : m(u_j) = \max_{1 \leq l \leq k} m(u_l)\}$
16: $\quad\quad$ **se** $m(u_{max}) \leq \left\lceil \frac{m_i}{2} \right\rceil$ **então**
17: $\quad\quad\quad bw(v_i) \leftarrow \left\lceil \frac{m_i}{2} \right\rceil$
18: $\quad\quad$ **senão**
19: $\quad\quad\quad bw(v_i) \leftarrow bw(u_{max})$
20: $\quad\quad$ **fim se**
21: $\quad$ **fim se**
22: **fim para**
23: **retorne** $bw(v_n)$

4.4 Cografos

Nesta seção será apresentado um algoritmo linear para solução do problema de BANDWIDTH na classe de cografos.

Como dito na seção anterior, o resultado de BANDWIDTH para cografos estende o obtido anteriormente para *quasi-threshold*. E esse feito deve-se à demonstração, no artigo [35], de como calcular em tempo $O(1)$ o resultado de *bandwidth* para uma junção de dois grafos. Esse era um resultado buscado desde 1988, quando Chung, num artigo intitulado "*Labelings of graphs*", encontrou o resultado de BANDWIDTH para a junção de alguns grafos em particular. Nos anos seguintes, em 1991 e em 1994, alguns limites foram estabelecidos para o problema, por Liu, Wang e Williams; e por Lai, Liu e Williams, respectivamente. Mas só em 1997, por Yan, é que o cálculo dessa operação foi resolvido por completo.

Definição (Cografo)**.** *Cografos são definidos como segue.*

1. *um grafo com exatamente um vértice é um cografo;*
2. *se G é um cografo, então $\overline{G}$ também é;*
3. *se G e H são cografos, então $G \cup H$ também é;*

Com o resultado da equação (1.1), pode-se substituir a regra "*2.*" pela seguinte regra.

2'. se G e H são cografos, então $G + H$ também é.

Em 1985, Corneil *et al* apresentaram um algoritmo linear para a construção da coárvore de um cografo dado como entrada. Tal construção possibilitou o desenvolvimento de algoritmos polinomiais, quando restrito a cografos, para problemas intratáveis em grafos em geral. Os problema de número cromático, isomorfismo, clique, *minimum fill-in* são alguns exemplos desses casos. Para o problema de BANDWIDTH, a construção da coárvore também possibilitou a construção de um algoritmo rápido, que usa uma busca em profundidade e as equações (2.8) e (2.9).

Inicialmente o algoritmo constrói a coárvore de um cografo de entrada e, então, executa uma busca em profundidade nela, contabilizando, para cada nó interior, o *bandwidth* da operação do nó sobre os filhos: junção ou união. A contabilização da junção e da união é feita com as equações (2.8) e (2.9). Como a busca é linear, a

construção da coárvore é linear e o cálculo da junção ou da união é $O(1)$, o algoritmo final é linear. Pela importância do resultado da equação (2.9) para esta solução, a sua demonstração será vista neste momento.

Teorema 4.7. *Sejam G e H dois grafos disjuntos. O $bw(G+H)$ é dado por* $bw(G+H) = \min\{\quad \max\{bw(G), \left\lceil \frac{|V(G)|}{2} - 1 \right\rceil\} + V(H) \quad , \quad \max\{bw(H), \left\lceil \frac{|V(H)|}{2} - 1 \right\rceil\} + V(G) \quad \}$

Demonstração. Sejam G e H dois grafos disjuntos tal que $V(G) = \{u_1, \ldots, u_n\}$ e $V(H) = \{v_1, \ldots, v_m\}$. E seja g uma rotulação ótima de G.

Define-se uma nova rotulação f para $G + H$ como segue.

$$f(w) = \begin{cases} g(w) & \text{se } w \in V(G) \text{ e } 1 \leq g(w) \leq \left\lceil \frac{n}{2} \right\rceil \\ j + \left\lceil \frac{n}{2} \right\rceil & \text{se } w = v_j, \text{ com } w \in V(H) \\ g(w) + m & \text{se } w \in V(G) \text{ e } \left\lceil \frac{n}{2} \right\rceil + 1 \leq g(w) \leq n \end{cases}$$

A nova rotulação pode ser vista graficamente na figura 4.11, onde cada rotulação de G, H e $G + H$ é representada por uma reta numerada, na qual cada vértice do respectivo grafo ocupa a posição correspondente ao seu rótulo.

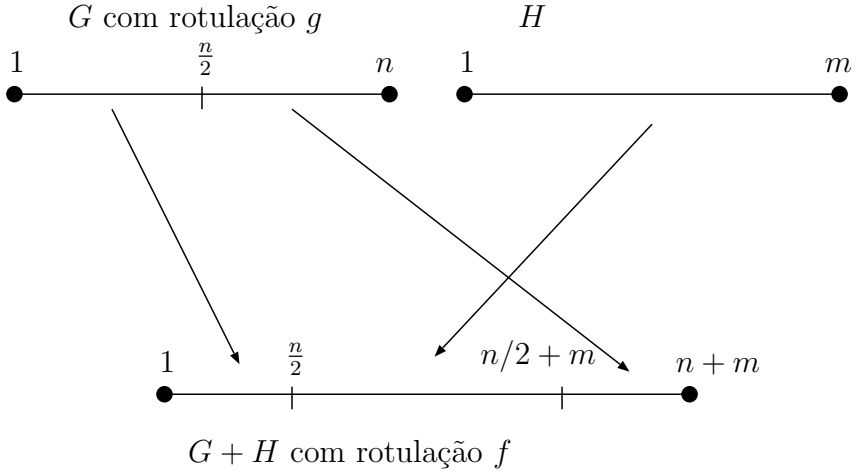

Figura 4.11: Nova rotulação f de $G + H$ em função das rotulações anteriores de G e de H.

Como, na nova rotulação f, o grafo H sofreu apenas incremento no valor dos rótulos de todos os vértices, então o $bw_f(H)$ permanece o mesmo.

$$bw_f(H) = bw_g(H) \leq m - 1 \tag{4.4}$$

O grafo G, porém, teve sua rotulação alterada, ao separar o conjunto de vértices em dois. Suponha que $bw(G) \geq \lceil \frac{n}{2} \rceil$. Isso implica que existe ao menos uma aresta ligando os vértices separados na rotulação f. Então, $bw_f(G) = bw_g(G) + m = bw(G) + m$. Agora suponha que $bw(G) \leq \lceil \frac{n}{2} \rceil - 1$. Nesse caso, $bw_f(G) \not> \lceil \frac{n}{2} \rceil - 1 + m$, logo $bw_f(G) \leq \lceil \frac{n}{2} \rceil - 1 + m$. Essa análise de $bw_f(G)$ pode ser resumida na equação (4.5).

$$bw_f(G) = bw(G) + m \tag{4.5}$$

A maior distância dentre as novas arestas da junção é $\lceil \frac{n}{2} \rceil + m - 1$, logo pode-se dizer que

$$\max_{1 \leq i \leq n, 1 \leq j \leq m} \{|f(v_j) - f(u_i)|\} = \left\lceil \frac{n}{2} \right\rceil + m - 1 \tag{4.6}$$

Assim, a partir dos resultados (4.4), (4.5) e (4.6), obtém-se a inequação (4.7).

$$\begin{aligned} bw(G+H) \leq bw_f(G+H) &= \max\{bw(G) + m, \left\lceil \frac{n}{2} \right\rceil + m - 1\} \\ bw(G+H) &\leq \max\{bw(G), \left\lceil \frac{n}{2} \right\rceil - 1\} + m \end{aligned} \tag{4.7}$$

Analogamente, por uma rotulção f' de $H + G$ tal que

$$f'(w) = \begin{cases} g(w) & \text{se } w \in V(H) \text{ e } 1 \leq g(w) \leq \lceil \frac{n}{2} \rceil \\ j + \lceil \frac{n}{2} \rceil & \text{se } w = v_j \\ g(w) + m & \text{se } w \in V(H) \text{ e } \lceil \frac{n}{2} \rceil + 1 \leq g(w) \leq n \end{cases}$$

obtem-se a inequação (4.8).

$$bw(G+H) \leq \max\{bw(H), \left\lceil \frac{m}{2} \right\rceil - 1\} + n \tag{4.8}$$

Por outro lado, considere uma rotulação ótima l de $G+H$. Se o rótulo 1 pertence ao grafo G e o rótulo $m+n$, ao grafo H (ou vice-versa), a junção de G e H implica uma aresta entre $l^{-1}(1)$ e $l^{-1}(m+n)$, e, assim, $bw(G+H) = m+n-1$, caso em que $G+H$ é um grafo completo e, portanto, G e H são completos. Mais adiante será visto que o resultado desse caso particular está de acordo com a equação da rotulação ótima para $G+H$.

Considere, a partir de agora, que os rótulos 1 e $m+n$ sejam dados ao mesmo grafo, G ou H. O resultado a seguir supõe que esses rótulos pertencem ao grafo G, mas deve ser avaliado também o caso em que pertencem a H. O resultado

final da rotulação ótima de *bandwidth*, portanto, deve ser o mínimo entre as duas possibilidades.

O $bw(G+H)$ é descrito por

$$\begin{aligned} bw(G+H) &= bw_l(G+H) \\ &= \max_{w_iw_j \in E(G+H)} \{|l(w_i) - l(w_j)|\} \\ &= \max\{bw_l(G), bw_l(H), \max_{1\leq i\leq n, 1\leq j\leq m} \{|l(u_i) - l(v_j)|\}\} \end{aligned}$$

É importante notar três fatos para a solução deste caso.

Fato 1. $bw_l(H) < \max_{1\leq i\leq n,1\leq j\leq m}\{|l(u_i) - l(v_j)|\}$ porque ambos os vértices extremos de $G+H$ — $l^{-1}(1)$ e $l^{-1}(m+n)$ —, que pertencem a G, são adjacentes aos extremos da aresta v_iv_j de H que possui diferença de rótulos igual a $bw_l(H)$. Então, pela figura 4.12, pode-se ver que $|l(v_j) - 1| > bw_l(H) = |l(v_j) - l(v_i)|$ (o mesmo ocorre com o extremo $l^{-1}(m+n)$, mas apenas a análise de um dos extremos é necessária).

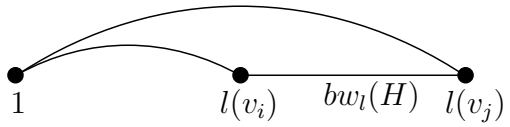

Figura 4.12: Exemplificação de que $bw_l(H) < \max_{1\leq i\leq n,1\leq j\leq m}\{|l(u_i) - l(v_j)|\}$.

Fato 2. $\max_{1\leq i\leq n,1\leq j\leq m}\{|l(u_i) - l(v_j)|\} \geq \lceil \frac{n}{2} \rceil + m - 1$.

Suponha $v \in H$ o vértice de H com menor rótulo em l, tal que $l(v) \leq \lfloor \frac{n}{2} \rfloor + 1$. Nesse caso, a aresta $\{v, l^{-1}(m+n)\}$ confere $\max_{1\leq i\leq n,1\leq j\leq m}\{|l(u_i) - l(v_j)|\} \geq \lceil \frac{n}{2} \rceil + m - 1$. Por outro lado, suponha que $l(v) > \lfloor \frac{n}{2} \rfloor + 1$. Como v é o primeiro vértice de H, e H tem m vértices, então existe um vértice x de H que possui rótulo $l(x) \geq \lfloor \frac{n}{2} \rfloor + 2 + m$, tal que a aresta $\{l^{-1}(1), v\}$ confere $\max_{1\leq i\leq n,1\leq j\leq m}\{|l(u_i) - l(v_j)|\} \geq \lfloor \frac{n}{2} \rfloor + m + 2 - 1 \geq \lceil \frac{n}{2} \rceil + m - 1$.

Fato 3. Sejam u_i e u_j vértices de G tal que $|l(u_j) - l(u_i)| = bw_l(G)$, e considere, sem perda de generalidade, que $l(u_i) < l(u_j)$. Se houver algum vértice $v \in V(H)$ de rótulo $l(v) < l(u_i)$ ou $l(v) > l(u_j)$, então $\max_{1\leq i\leq n,1\leq j\leq m}\{|l(u_i) - l(v_j)|\} > bw_l(G)$. Caso contrário, se todo $v \in V(H)$ tiver $l(u_i) < l(v) < l(u_j)$, então $bw_l(G) \geq bw(G) + m$ pois há m vértices de H entre u_i e u_j.

Logo,

$$\begin{aligned} bw(G+H) &= \max\{bw_l(G), bw_l(H), \max_{1\leq i\leq n, 1\leq j\leq m}\{|l(u_i)-l(v_j)|\}\} \\ &= \max\{bw_l(G), \max_{1\leq i\leq n, 1\leq j\leq m}\{|l(u_i)-l(v_j)|\}\} \text{ (pelo Fato 1.)} \\ &\geq \max\{bw(G)+m, \left\lceil\frac{n}{2}\right\rceil + m - 1\} \text{ (pelo Fato 1 e Fato 2.)} \\ &\geq \max\{bw(G), \left\lceil\frac{n}{2}\right\rceil - 1\} + m \text{ (simplificando)} \end{aligned}$$

Como dito,

$$bw(G+H) \geq \min\{\max\{bw(G), \left\lceil\frac{n}{2}\right\rceil - 1\} + m; \max\{bw(H), \left\lceil\frac{m}{2}\right\rceil - 1\} + n\} \quad (4.9)$$

Dos resultados de (4.7), (4.8) e (4.9), obtém-se o resultado final da equação (2.9). □

O algoritmo 5 calcula o *bandwidth* de um cografo de entrada, em tempo linear.

Algoritmo 5 Calcular o *bandwidth* de um cografo, em $O(n)$.

Entrada: Um cografo $G = (V, E)$.
Saída: $bw(G)$.

Encontrar a coárvore T_G de G
Rotular os vértices de T_G em $1, \ldots, |V(T_G)|$ por um percurso em pós-ordem
para $i = 1, \ldots, |V(T_G)|$ **faça**
    $b(v_i) \leftarrow 0$
    $p(v_i) \leftarrow 1$
fim para
para $i = 1, \ldots, |V(T_G)|$ **faça**
    se v_i não é folha **então**
        Sejam $u_1, \ldots, u_k$ os filhos de v_i
        se rótulo de v_i for 0 **então**
            $p(v_i) \leftarrow \sum_{1 \leq j \leq k} p(u_j)$
            $b(v_i) \leftarrow \max\{b(u_1), \ldots, b(u_k)\}$
        senão
            $p(v_i) \leftarrow p(u_1) + p(u_2)$
            $x \leftarrow \max\{b(u_1), \lceil \frac{p(u_1)}{2} - 1 \rceil\} + p(u_2)$
            $y \leftarrow \max\{b(u_2), \lceil \frac{p(u_2)}{2} - 1 \rceil\} + p(u_1)$
            $b(v_i) \leftarrow \min\{x, y\}$
            para $u = u_3, \ldots, u_k$ **faça**
                $p(v_i) \leftarrow p(v_i) + p(u)$
                $x \leftarrow \max\{b(v_i), \lceil \frac{p(v_i)}{2} - 1 \rceil\} + p(u)$
                $y \leftarrow \max\{b(u), \lceil \frac{p(u)}{2} - 1 \rceil\} + p(v_i)$
                $b(v_i) \leftarrow \min\{x, y\}$
            fim para
        fim se
    fim se
fim para
retorne $b(v_{|V(T_G)|})$

4.5 P_4-esparso

Nesta seção será apresentado um algoritmo linear para solução do problema de BANDWIDTH na classe de grafos P_4-esparso.

P_4-esparso é uma superclasse de cografo. Cografo é a classe dos grafos que não possuem P_4 como subgrafo induzido. De forma mais abrangente, P_4-esparso é o conjunto dos grafos tal que em todo conjunto de cinco vértices existe no máximo um P_4.

Sabe-se que para um cografo é possível encontrar a sua coárvore, única, em tempo linear, na qual os nós internos da árvore são operações de junção e união. Assim como cografo, a classe P_4-esparso também admite a caracterização de um grafo por uma árvore, neste caso chamada *ps-tree*, na qual além das operações de união e junção existe uma nova operação: ω, definida por Jamison e Olariu [16], em 1992. E a construção da *ps-tree* também é linear; algoritmo este apresentado em 1992 também por Jamison e Olariu [15].

Com o algoritmo linear para a construção da *ps-tree* e o resultado do teorema 4.7 além da equação (2.8), resta apenas calcular o valor de *bandwidth* para a nova operação ω, para que seja possível construir um algoritmo linear que resolva BANDWIDTH para P_4-esparso. Este algoritmo assim como em cografos executa um percurso em pós-ordem na *ps-tree* de um grafo P_4-esparso, contabilizando, no percurso da raiz de cada sub-árvore, o valor do *bandwidth* em questão. Os teoremas 4.8 e 4.9 a seguir apresentam esses resultados. A classe P_4-esparso e a nova operação ω serão definidos abaixo.

Definição (Operação binária ω)**.** *Sejam dois grafos* $G_1 = \{V_1, \emptyset\}$ *e* $G_2 = \{V_2, E_2\}$ *com* $V_2 = \{v\} \cup K \cup R$ *tal que*

- K *é uma clique com* $|K| = |V_1| + 1$*;*
- *cada vértice de* R *é adjacente a todos os vértices de* K *mas não é adjacente a* v*;*
- *a vizinhança de* v *pode ser de dois tipos para algum* $u \in K$*:*
  1. $N(v) = \{u\}$
  2. $N(v) = K \setminus \{u\}$

Seja f uma bijeção entre V_1 e $K \setminus \{u\}$. Define-se $G_1 \omega G_2$ com $V = V_1 \cup V_2$ e E conforme a seguir.

1. *Se $N(v) = \{u\}$, então $E = E_2 \cup \{wf(w) : w \in V_1\}$*
2. *Se $N(v) = K \setminus \{u\}$, então $E = E_2 \cup \{wz : w \in V_1 \text{ e } z \in K \setminus \{f(w)\}\}$*

Definição (P_4-esparso)**.** *P_4-esparso são definidos como segue.*

1. *um grafo com exatamente um vértice é um grafo P_4-esparso;*
2. *se G é um grafo P_4-esparso, então $\overline{G}$ também é;*
3. *se G e H são grafos P_4-esparso, então $G \cup H$ também é;*
4. *se G e H são grafos P_4-esparso, então $G \omega H$ também é;*

Com o resultado da equação (1.1), pode-se substituir a regra "*2.*" pela seguinte regra.

2'. se G e H são grafos P_4-esparso, então $G + H$ também é.

Sejam G_1 e G_2 grafos como apresentados na definição da operação ω, e considere $G = G_1 \omega G_2 = (I \cup K \cup R, E)$ com

- $I = V_1 \cup \{v\}$ com $|I| = |K| = n$,
- $I = \{w_1, \ldots, w_n\}$ é um conjunto independente, $K = \{u_1, \ldots, u_n\}$ é uma clique e $R = \{v_1, \ldots, v_m\}$ com $G_2[R] = H$ e $|R| = m$,
- se a bijeção de I para K for do tipo 1, segundo a definição da operação ω, representa-se o grafo G por $\omega(n, H, 1)$; se a bijeção for do tipo 2, $G = \omega(n, H, 2)$.

A figura 4.13 apresenta um exemplo de rotulação de um grafo do tipo 1, para facilitar o entendimento do teorema 4.8. À esquerda da figurada, o grafo possui todas as arestas; na parte da direita, porém, só estão representadas as arestas de cada vértice que definem o *bandwidth* local para aquele vértice.

Teorema 4.8. $bw(\omega(n, H, 1)) = \max\left\{bw(H), \left\lceil \frac{m}{2} \right\rceil - 1\right\} + n$

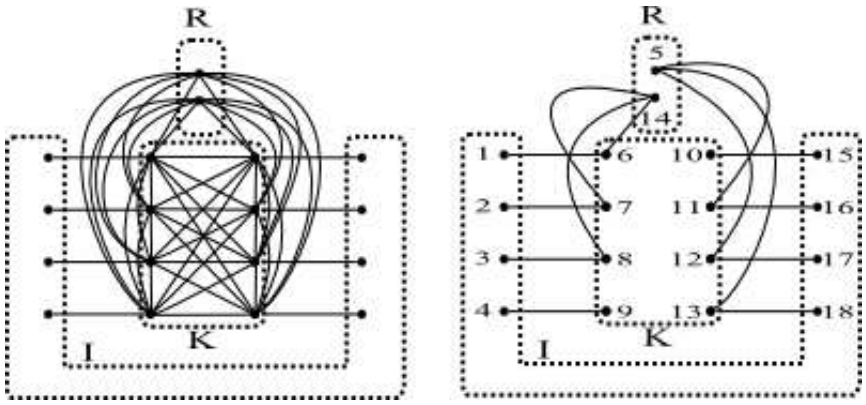

Figura 4.13: Exemplo de rotulação ótima de um grafo do tipo 1.

Demonstração. Seja g uma rotulação ótima de H, e considere uma nova rotulação f de G definida abaixo e representada na figura 4.14.

$$f(v) = \begin{cases} i & \text{se } v = w_i \text{ e } 1 \leq i \leq \left\lfloor \frac{n}{2} \right\rfloor \\ i + \left\lfloor \frac{n}{2} \right\rfloor & \text{se } v = v_i \text{ e } 1 \leq i \leq \left\lceil \frac{m}{2} \right\rceil \\ i + \left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{m}{2} \right\rceil & \text{se } v = u_i \\ i + \left\lfloor \frac{n}{2} \right\rfloor + n & \text{se } v = v_i \text{ e } \left\lceil \frac{m}{2} \right\rceil + 1 \leq i \leq m \\ i + n + m & \text{se } v = w_i \text{ e } \left\lfloor \frac{n}{2} \right\rfloor + 1 \leq i \leq n \end{cases}$$

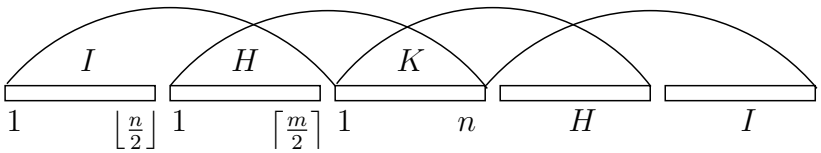

Figura 4.14: Rotulação de um grafo P_4-esparso com as arestas mais importantes para o problema BANDWIDTH destacadas.

Na figura 4.14 estão representadas as arestas mais importantes para os critérios de cálculo de $bw(\omega(n, H, 1))$, que será apresentado a seguir.

$$
\begin{aligned}
bw(\omega(n,H,1)) &\leq \max_{xy\in E(\omega(n,H,1))} |f(x)-f(y)| \\
&\leq \max\left\{bw_f(H), \max_{x\in K, y\in K\cup R}|f(x)-f(y)|, \max_{1\leq i\leq n}|f(u_i)-f(w_i)|\right\} \\
&\leq \max\left\{bw_g(H)+n, \left\lceil\frac{m}{2}\right\rceil - 1 + n, \left\{\left\lfloor\frac{n}{2}\right\rfloor + \left\lceil\frac{m}{2}\right\rceil, \left\lfloor\frac{m}{2}\right\rfloor + \left\lceil\frac{n}{2}\right\rceil\right\}\right\} \\
&\leq \max\left\{bw(H), \left\lceil\frac{m}{2}\right\rceil - 1\right\} + n \qquad (4.10)
\end{aligned}
$$

E como $K_n + H$ é subgrafo de $\omega(n, H, 1)$, pelo limite inferior (2.6), tem-se:

$$
\begin{aligned}
bw(\omega(n,H,1)) &\geq bw(K_n + H) \\
&\geq \min\left\{\max\left\{bw(K_n), \left\lceil\frac{n}{2}\right\rceil - 1\right\} + m, \max\left\{bw(H), \left\lceil\frac{m}{2}\right\rceil - 1\right\} + n\right\} \\
&\geq \max\left\{bw(H), \left\lceil\frac{m}{2}\right\rceil - 1\right\} + n \qquad (4.11)
\end{aligned}
$$

Os resultados das inequações (4.10) e (4.11) concluem que

$$
bw(\omega(n,H,1)) = \max\left\{bw(H), \left\lceil\frac{m}{2}\right\rceil - 1\right\} + n
$$

□

A figura 4.15 apresenta um exemplo de rotulação de um grafo do tipo 2, para facilitar o entendimento do teorema 4.9. À esquerda da figurada, o grafo possui todas as arestas; na parte da direita, porém, só estão representadas as arestas de cada vértice que definem o *bandwidth* local para aquele vértice.

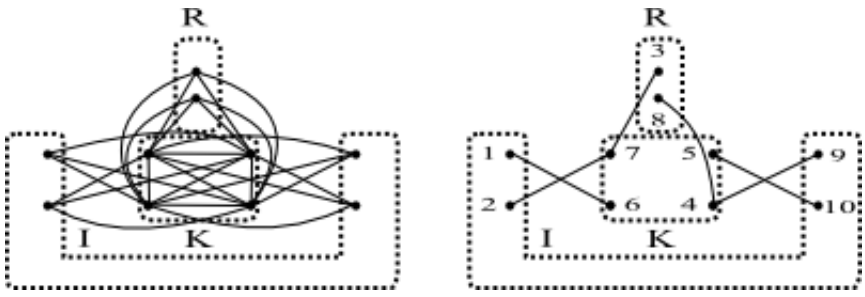

Figura 4.15: Exemplo de rotulação ótima de um grafo do tipo 2.

Teorema 4.9. $bw(\omega(n,H,2)) = \max\left\{bw(H), \left\lceil\frac{m+n}{2}\right\rceil - 2\right\} + n$

Demonstração.

$$f(v) = \begin{cases} i & \text{se } v = w_i \text{ e } 1 \leq i \leq \left\lfloor \frac{n}{2} \right\rfloor \\ i + \left\lfloor \frac{n}{2} \right\rfloor & \text{se } v = v_i \text{ e } 1 \leq i \leq \left\lceil \frac{m}{2} \right\rceil \\ n - i + 1 + \left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{m}{2} \right\rceil & \text{se } v = u_i \\ i + \left\lfloor \frac{n}{2} \right\rfloor + n & \text{se } v = v_i \text{ e } \left\lceil \frac{m}{2} \right\rceil + 1 \leq i \leq m \\ i + n + m & \text{se } v = w_i \text{ e } \left\lfloor \frac{n}{2} \right\rfloor + 1 \leq i \leq n \end{cases}$$

$$\begin{aligned} bw(\omega(n,H,2)) &\leq \max_{xy \in E(\omega(n,H,2))} |f(x) - f(y)| \\ &\leq \max\left\{ bw_f(H), \max_{x \in K, y \in K \cup R} |f(x) - f(y)|, \max_{1 \leq i \neq j \leq n} |f(u_i) - f(w_i)| \right\} \\ &\leq \max\left\{ bw_g(H) + n, \left\lceil \frac{m}{2} \right\rceil - 1 + n, \left\{ \left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{m}{2} \right\rceil + n - 2, \left\lfloor \frac{m}{2} \right\rfloor + \left\lceil \frac{n}{2} \right\rceil + n - 2 \right\} \right\} \\ &\leq \max\left\{ bw(H), \left\lceil \frac{m+n}{2} \right\rceil - 2 \right\} + n \end{aligned} \tag{4.12}$$

E como $K_n + H$ é subgrafo de $\omega(n, H, 2)$, pelo limite inferior (2.6), tem-se:

$$\begin{aligned} bw(\omega(n,H,2)) &\geq bw(K_n + H) \\ &\geq \min\left\{ \max\left\{ bw(K_n), \left\lceil \frac{n}{2} \right\rceil - 1 \right\} + m, \max\left\{ bw(H), \left\lceil \frac{m}{2} \right\rceil - 1 \right\} + n \right\} \\ &\geq \max\left\{ bw(H), \left\lceil \frac{m}{2} \right\rceil \right\} + n \\ &\geq bw(H) + n \end{aligned} \tag{4.13}$$

É necessário encontrar o limite inferior $\left\lceil \frac{m+n}{2} \right\rceil + n - 2$ para que o valor de $bw(\omega(n, H, 2)$ seja estabelecido.

Considere l uma rotulação ótima de G e sejam x_1, x_2, y_1 e $y_2 \in I \cup R$ com $l(x_1) \leq l(x_2) \leq l(y_1) \leq l(y_2)$, e $z \in I \cup R \setminus \{x_1, x_2, y_1, y_2\}$ com $l(x_2) \leq l(z) \leq l(y_1)$. Sejam também u_i e $u_j \in K$ com $l(u_i) \leq l(u_j)$.

Uma das duas arestas apresentadas na figura 4.16 terá diferença de rótulo dos vértice como o limite inferior desejado. Para isso, serão analisados os casos em que u_n tem rótulo menor que o rótulo de y_1 e em que tem rótulo maior. Os quatro casos possíveis estão listados na figura 4.17, na qual os valores a, b e c são naturais que satisfazem $a + b + c + 4 = m + 2n - 4$.

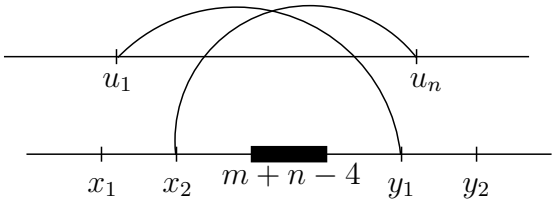

Figura 4.16: Arestas de $\omega(n, H, 2)$ que garantem o limite infeior $\lceil \frac{m+n}{2} \rceil + n - 2$.

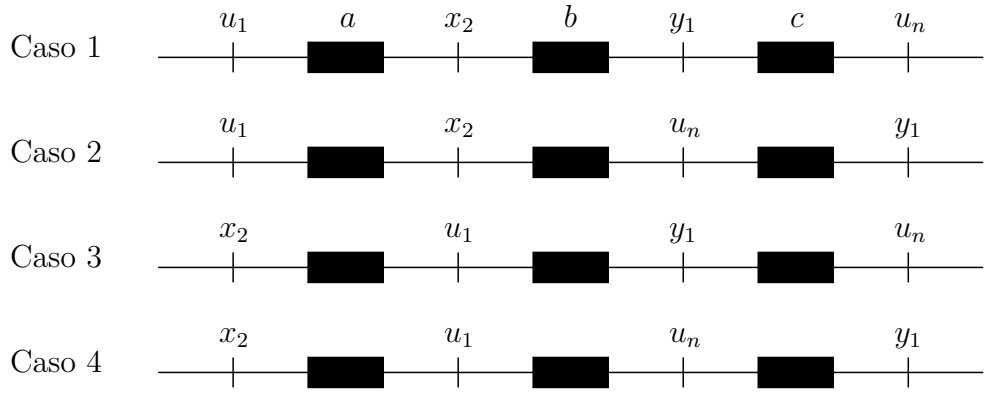

Figura 4.17: Possíveis casos para a distribuição dos vértices x_1, x_2, y_1, y_2, u_1 e u_n.

Para a análise dos casos a seguir, supõe-se que $l(u_n) - l(x_2) < n - 2 + \lceil \frac{m+n}{2} \rceil$. Se esse diferença de rótulo não ocorrer, obviamente não é necessária análise.

Caso 1: Conclui que $l(y_1) - l(u_1) > \lceil \frac{m+n}{2} \rceil + n - 2$

$$\begin{aligned} l(y_1) - l(u_1) &= a + b + 2 \\ \left\lceil \frac{m+n}{2} \right\rceil + n - 2 &> b + c + 2 \\ b &\geq m + n - 4 \\ a + b + c + 4 &= m + 2n - 2 \end{aligned}$$

Caso 2: Conclui que $l(y_1) - l(u_1) > \lceil \frac{m+n}{2} \rceil + n - 2$

$$\begin{aligned} l(y_1) - l(u_1) &= a + b + c + 3 \\ a + b + c + 4 &= m + 2n - 2 \end{aligned}$$

Caso 3: Absurdo!

$$\begin{aligned} l(y_1) - l(u_1) &= b + 1 \\ a + b + c + 3 &< \left\lceil \frac{m+n}{2} \right\rceil + n - 2 \\ a + b + c + 4 &= m + 2n - 2 \end{aligned}$$

Caso 4: Conclui que $l(y_1) - l(u_1) > \left\lceil \frac{m+n}{2} \right\rceil + n - 2$

$$\begin{aligned} l(y_1) - l(u_1) &= b + c + 2 \\ b + 1 &> n \\ \left\lceil \frac{m+n}{2} \right\rceil + n - 2 &> a + b + 2 \\ a + b + c + 4 &= m + 2n - 2 \end{aligned}$$

Dos resultados anteriores, conclui-se a afirmação deste teorema. □

4.6 *Chain graphs*

A classe de grafos *chain* [21] possui um algoritmo $O(n^2 \log n)$ para resolver o problema BANDWIDTH.

Chain é uma subclasse de bipartidos em que os vértice de cada classe de cor têm uma relação de inclusão entre as vizinhanças como definido a seguir.

Definição (Grafo *chain*)**.** *Um grafo bipartido $G = (X, Y, E)$ é um grafo chain se existe uma ordenação dos vértices de $X = \{x_1, \ldots, x_p\}$, tal que*

$$N(x_1) \supseteq N(x_2) \supseteq \ldots \supseteq N(x_p)$$

É fácil verificar que essa característica de inclusão entre as vizinhanças dos vértices do conjunto X implica também uma ordenação dos vértices do conjunto $Y = \{y_1, \ldots, y_q\}$ tal que

$$N(y_1) \subseteq N(y_2) \subseteq \ldots \subseteq N(y_q)$$

De fato, seja y_i um vértice qualquer de Y, e x_a o vértice de X de maior índice e adjacente a y_i: $N(y_i) = \{x_1, \ldots, x_a\}$. Analogamente, sejam y_j e x_b, respectivamente, um vértice qualquer de Y e o vértices de X com maior índice adjacente a y_j: $N(y_j) = \{x_1, \ldots, x_b\}$. Em função dos índices a e b, obtém-se $N(y_i) \subseteq N(y_j)$ ou $N(y_j) \subseteq N(y_i)$, definindo uma ordenação total no conjunto Y.

Para os próximos parágrafos, considere $G = (X, Y, E)$ um grafo *chain* conexo, tal que $N(x_1) = Y$ e $N(y_q) = X$.

Definição. *Sejam $H_0, H_1, \ldots, H_{p-1}$ os seguintes grafos. O grafo H_0 é obtido de G ao se fazer o conjunto X uma clique. E, para $i = 1, \ldots, p-1$, seja l o menor índice tal que x_{i+1} é adjacente a y_l. O grafo H_i é obtido de G ao fazer uma clique em $\{x_1, \ldots, x_i\} \cup \{y_l, \ldots, y_q\}$.*

Uma consequência imediata da definição dos grafos $H_0, H_1, \ldots, H_{p-1}$ é que qualquer um deles é um supergrafo de G: $G \subseteq H_i$ para $0 \leq i \leq p-1$.

Definição (Triangulação)**.** *Um grafo cordal H é chamado uma triangulação de um grafo G se H e G têm o mesmo conjunto de vértices, e G é subgrafo de H.*

Teorema 4.10. *Os grafos $H_0, H_1, \ldots, H_{p-1}$ definidos anteriormente são todos grafos de intervalo.*

Demonstração. Duas provas serão dadas para esse teorema. A primeira será na forma do modelos de intervalo para os grafos $H_0, H_1, \ldots, H_{p-1}$ que podem ser vistos nas figuras 4.18 e 4.19.

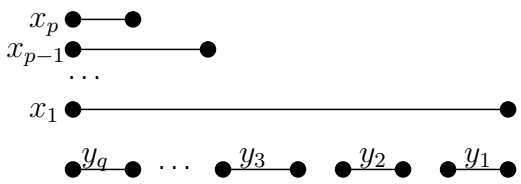

Figura 4.18: Exemplificação do modelo de intervalo do grafo H_0.

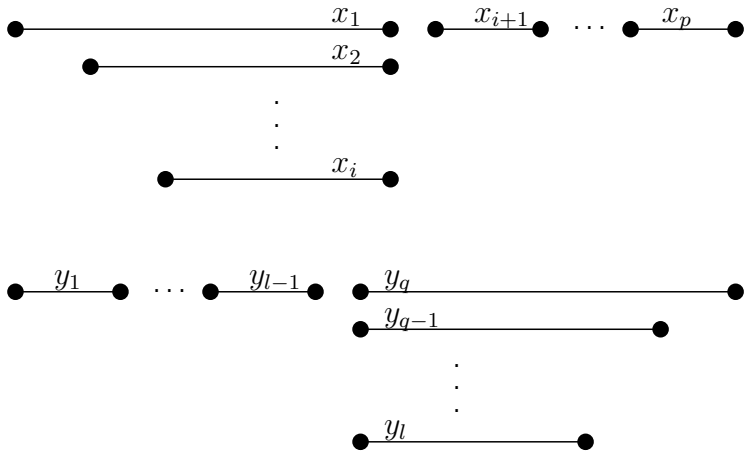

Figura 4.19: Exemplificação do modelo de intervalo dos grafos $H_1, H_2 \ldots, H_{p-1}$.

A segunda prova será feita usando uma equivalência entre grafo de intervalo e grafo cordal que não contém tripla asteroidal. A demonstração de que o grafo H_0 é cordal, vem do fato dele ser *split* (clique em X), que é cordal. E não pode haver tripla asteroidal porque qualquer conjunto independente de três vértices será subconjunto de Y, e, em função da inclusão das vizinhanças, qualquer caminho entre quaisquer dois deles será adjacente ao terceiro.

Assim, será mostrado inicialmente que os grafo $H_1, \ldots, H_{p-1}$ são de intervalo são cordais, e posteriormente que eles não possuem tripla asteroidal.

Seja $i \geq 1$ e seja l o menor índice para o qual x_{i+1} é adjacente a y_l. Pela definição, $\{x_1, \ldots, x_i\} \cup \{y_l, \ldots, y_q\}$ é uma clique, e por G ser bipartido $\{x_{i+1}, \ldots, x_p\} \cup \{y_1, \ldots, y_{l-1}\}$ é um conjunto independente, logo G é um grafo *split*, que é cordal.

Suponha por contradição que H_i tenha uma tripla asteroidal AT x, y e z. Como AT é um conjunto independente, então, para os casos em que há dois vértices no conjunto $\{x_{i+1}, \ldots, x_p\}$, ou dois no conjunto $\{y_1, \ldots, y_{l-1}\}$, a restrição do caminho P não será satisfeita. Considere, sem perda de generalidade, x e y pertencentes a $\{x_{i+1}, \ldots, x_p\}$, de forma que $N(x) \subseteq N(y)$, então todo caminho P entre x e z terá ao menos um vértice de P adjacente a y. A única outra possibilidade de distribuição dos vértices entre os conjuntos é, sem perda de generalidade, $x \in \{x_{i+1}, \ldots, x_p\}$, $y \in \{y_l, \ldots, y_q\}$ e z na clique $(\{x_1, \ldots, x_i\} \cup \{y_l, \ldots, y_q\})$. Mas nesse formato todo caminho entre x e y deve passar por S, que contém um vizinho de z. □

Como todos $H_0, H_1, \ldots, H_{p-1}$ são supergrafos de G, e também grafos de intervalo (cordal, de forma mais ampla), então são triangulações de G.

Teorema 4.11. *Toda triangulação H de G tem um dos grafos $H_0, H_1, \ldots, H_{p-1}$ como um subgrafo.*

Demonstração. Se X induz uma clique em H, então H_0 é subgrafo de H, e se Y induz uma clique em H, então H_1 é subgrafo de H. Considere que nenhum desses dois casos ocorre.

Sejam k máximo e t mínimo tal que $\{x_1, \ldots, x_k\}$ e $\{y_t, \ldots, y_q\}$ induzam cliques em H. Afirma-se que $x_{k+1}y_{t-1} \notin E(G)$. De fato, se $x_{k+1}y_{t-1} \in E(G)$, o subgrafo bipartido $(\{x_1, \ldots, x_{k+1}\}, \{y_{t-1}, \ldots, y_q\})$ seria completo (bipartido completo), e uma triangulação de um grafo bipartido completo é tal que uma das classes de cor é uma clique, que induziria a um ciclo induzido de tamanho maior ou igual a 4 em H: absurdo.

Seja, então, l o índice mínimo tal que x_{k+1} e y_l sejam adjacentes; $l \geq t$. O que prova que H_k é subgrafo de H. □

Corolário 4.3. *O conjunto de todas as triangulações minimais de G está contido em $\{H_0, H_1, \ldots, H_{p-1}\}$.*

Demonstração. Se, para toda triangulação H de G, existe H_i tal que $H \supseteq H_i$, então $H \supseteq H_i \supseteq G$. E sendo H minimal, conclui-se que $H = H_i$. □

Corolário 4.4. *Para todo grafo G (conexo), existe um grafo H de intervalo próprio tal que $G \subseteq H$ com $V(G) = V(H)$.*

Demonstração. Seja $T = (F_V, W)$ a decomposição em caminho própria de G, na qual $F_V = \{X_1, \ldots, X_l\}$. Os elementos X_i, $1 \leq i \leq l$, estão dispostos nessa ordem no caminho da decomposição, que podemos representar por uma reta rotulada de 1 a l. Para cada vértice $v \in V(G)$, contido nos elementos $\{H_i, \ldots, H_j\}$ de F_V, defini-se I_v como um intervalo na reta de rotulação que inicia em i e termina em j. Tal grafo de intervalos gerado é o grafo H. □

Teorema 4.12. *Existe um algoritmo $O(n^2 \log n)$ em tempo para computar o $bw(G)$, sendo G um grafo chain.*

Demonstração. Seja H um supergrafo de intervalo próprio de G tal que $bw(G) = \omega(H) - 1$. Tal supergrafo H existe baseado nos resultados do lemas 2.3 e 2.2.

Como H é supergrafo de G com $V(G) = V(H)$, e H é de intervalo próprio, que é cordal, então H é uma triangulação de G. Pelo teorema 4.11, algum H_i, $0 \leq i \leq p-1$, seja H_k tal grafo, é subgrafo de H. Assim, $bw(H_k) \leq bw(H)$. E, pelas relações de inclusão entre os grafos, $bw(G) \leq bw(H_k) \leq bw(H)$. Como H é de intervalo próprio (ou intervalo unitário), $bw(H) = \omega(H)-1$ pelo corolário 4.2, e $\omega(H)-1 = bw(G)$ pela escolha de H. Unindo os resultados anteriores, $bw(G) \leq bw(H_k) \leq bw(H) = bw(G)$, logo $bw(G) = bw(H_k)$.

Resta saber qual H_k possui $bw(H_k) = bw(G)$. Será mostrado que tal H_k é aquele, dentre os H_i, com menor valor de $bw(H_i)$. Com efeito, sabe-se que $bw(G) \leq bw(H_i)$ para todo H_i porque $G \subseteq H_i$. Se por contradição $bw(G)$ for igual a algum $bw(H_k)$ que não seja o menor dentre os $bw(H_i)$, então o H_i de $bw(H_i)$ mínimo — seja H_l tal grafo — terá $bw(H_l) < bw(G)$: absurdo.

Então, o valor de $bw(G)$ pode ser calculado ao se calcular os valores de *bandwidth* de todos os grafos $H_0, H_1, \ldots, H_{p-1}$ e encontrar o menor deles. Os valores podem ser obtidos com o algoritmo $O(n \log n)$ para grafos de intervalo, de Sprague [33], resultando numa complexidade final de $O(n^2 \log n)$. □

Capítulo 5

Algoritmos exatos exponenciais para a classe geral de grafos

Este capítulo trata dos algoritmos exatos exponenciais para o problema BANDWIDTH na classe geral de grafos. Como pôde ser visto no capítulo 3 o problema é NP-completo para esta classe, então, a menos que $P = NP$, não há algoritmos polinomiais para resolver o problema; espera-se que exista tão somente algoritmos aproximativos ou exponenciais. Nesta dissertação não serão tratadas as soluções aproximadas, mas apenas as exatas, exluindo a solução óbvia de busca exaustiva, com complexidade $O(n^n)$.

5.1 Algoritmo polinomial para k fixo

O problema BANDWIDTH-K é uma variante do problema de decisão BANDWIDTH, em que o valor de k não é dado como parâmetro de entrada mas é fixo com a descrição do problema. Nesse caso, o único parâmetro dado é o grafo G a ser verificado se $bw(G) \leq k$.

Problema: BANDWIDTH-K
Instância: Um grafo G.
Questão: $bw(G) \leq k$?

O problema de determinar em tempo polinomial se um grafo G possui $bw(G) \leq k$ para um dado valor fixo de k — BANDWIDTH-K — foi inicialmente resolvido para $k = 2$ no artigo de Garey *et al* [6], onde é apresentado um algoritmo linear. Porém,

os outros casos de k, $k \geq 3$, ficaram sem resposta; em particular, na seção 10 desse mesmo artigo — "*Some open problems*" —, os autores questionaram se o problema seria NP-completo para $k = 3$: "*Is the problem 'Bandwidth(G)≤ 3' NP-complete, given an arbitrary graph (or perhaps a tree) G*?", esboçando uma expectativa de que o problema se tornaria difícil a partir do valor $k = 3$.

Entretanto, poucos anos depois (de 1978 para 1980), essa pergunta foi respondida negativamente por Saxe [31], onde foi apresentada uma família de algoritmos polinomiais para resolver o problema para qualquer k inteiro, sendo um algoritmo polinomial $O(n^{k+1})$ para cada valor possível de k. Esse limite superior foi refinado por Gurari [12] para $O(n^k)$, ao usar o mesmo algoritmo de Saxe, porém com estruturas de dados mais apropriadas.

Nesta seção será apresentada a família de algoritmos polinomiais para BANDWIDTH-K que foram desenvolvidos no artigo de Saxe [31].

5.1.1 Conceitos fundamentais

Definição (Extensão de rotulação)**.** *Uma rotulação g é dita extensão de f se o domínio de f, V_f, for subconjunto do domínio de g, V_g, de tal forma que $f(v) = g(v)$ para todo $v \in V_f$.*

Definição (Rotulação parcial provável)**.** *Uma rotulação parcial f é dita provável se for possível estendê-la a uma rotulação g tal que $bw_g(G) \leq k$.*

Definição (Arestas penduradas (*dangling edges*))**.** *Seja f um rotulação parcial de G. A aresta $uv \in E$ é dita pendurada de f se u está no domínio de f mas v não.*

Seja f uma rotulação parcial de G com domínio de tamanho p, $p < n$. Claramente, f não é provável a menos que:

1. $bw_f(G) \leq k$; e
2. sempre que $u, v \in V$ tal que $f(u) \leq p-k$ e $uv \in E$, v também está no domínio de f.

Se f satisfaz ambas as condições acima, então f é dita uma **rotulação parcial plausível**. Os últimos k vértices rotulados (ou menos, caso não haja k vértices ainda) — $f^{-1}(\max\{p-k+1, 1\}), \ldots, f^{-1}(p)$ — mais as arestas penduradas de f são chamados a **região ativa** de f.

O conceito fundamental para o desenvolvimento da família de algoritmos é a relação de equivalência definida no teorema 5.1. Essa relação particiona o conjunto de rotulações G em função das suas regiões ativas; o algoritmo, então, realiza uma busca em largura sobre essas regiões, com o auxílio da noção de sucessor de uma rotulação parcial, definida a seguir.

Definição (Sucessor de uma rotulação)**.** *Seja f uma rotulação parcial plausível de G. Uma sucessora de f é uma rotulação parcial plausível g que estende f por exatamente um vértice. Neste caso, a região ativa de g é dita sucessora da de f. Analogamente, diz-se que f (ou região ativa de f) é predecessora de g (ou região ativa de g).*

Teorema 5.1. *Sejam f e g duas rotulações parciais plausíveis de G com regiões ativas idênticas. Então:*

1. *f e g têm domínios idênticos;*

2. *f é provável se, e somente se, g é provável.*

Demonstração. Na primeira assertiva, como as regiões ativas são iguais, é necessário apenas se preocupar com os vértices do domínio de cada função com rótulo menor ou igual a $p - k$.

Um vértice v rotulado com $f(v) \leq p-k$ não possui vértice adjacente que não seja rotulado; e, mais que isso, todo caminho de um vértice v rotulado com $f(v) \leq p-k$ até um vértice da região ativa (com rótulo maior que $p - k$) deve incluir apenas vértices com rótulos menores ou iguais a $p - k$; caso contrário, em algum momento do caminho, a segunda condição para f ser plausível não seria satisfeita. É evidente então que os vértices do domínio de f são todos aqueles alcançáveis por caminho iniciados em vértices da região ativa e que não usem arestas penduradas de f (para não incluírem vértices não rotulados no caminho). Isso implica naturalmente que duas regiões ativas 'gerem' o mesmo domínio.

A segunda assertiva é facilmente demonstrável ao perceber que a mesma extensão pode ser aplicada a ambas as duas rotulações, uma vez que elas possuem mesma região ativa. □

5.1.2 O algoritmo

O algoritmo é essencialmente uma busca em largura sobre as regiões ativas do grafo, fazendo o uso da técnica de programação dinâmica.

Uma região ativa tem no máximo k vértices e cada vértice possui no máximo $2k$ arestas porque um grafo G com $bw(G) \leq k$ não tem vértice com grau maior que $2k$, por causa da inequação (2.5). Isso implica

$$\sum_{0 \leq i \leq k} \binom{n}{i} (i!)(2^{2k})^i = O(n^k)$$

classes de equivalência. Cada classe é representada por uma rotulação parcial pertencente a ela.

O algoritmo inicia o processamento com a classe de equivalência (uma classe é representada por uma rotulação parcial) que não possui nenhum vértice, *i.e.*, o conjunto vazio. A extensão das rotulações é feita iterativamente pelo acréscimo de um único vértice a cada passo do algoritmo, quando o número de vértices rotulados numa rotulação parcial cresce, mas enquanto o tamanho da região ativa permanece limitada a k vértices.

Para evitar a reanálise redundante de rotulações, usa-se a técnica de programação dinâmica. O pseudocódigo da família de algoritmos está descrito no algoritmo 6.

O algoritmo usa duas estruturas de dados:

1. uma fila Q, cujos elementos são as regiões ativas;
2. e um vetor A, que contém uma rotulação representante de cada região ativa do grafo.

Em cada $A[r]$, o campo *examined* ($A[r].examined$) é do tipo booleano e informa se a respectiva região ativa já foi analisada, e o campo *unplaced* ($A[r].unplaced$) contém uma lista com todos os vértices que não estão no domínio de cada rotulação parcial plausível com região ativa r.

Em virtude do número de regiões ativas e pelo fato de cada uma das regiões ser obtida da cabeça da fila Q na linha 2, o laço **repita-até** será executado $O(n^k)$. A linha 1, em particular, será executada exatamente $O(n^k)$. Desde que cada região ativa tem n sucessores, um para cada lista $A[r].unplaced$, a linha 3 será executada $O(n^{k+1})$ vezes. A análise das linhas de 4 a 13 é um pouco mais detalhada. É

Algoritmo 6 BANDWIDTH-K: verificar se um grafo G de entrada tem $bw(G) \leq k$, em tempo $O(n^{k+1})$.

Entrada: Um grafo $G = (V, E)$
Saída: **verdadeiro** $\Leftrightarrow bw(G) \leq k$

```
 1: repita
 2:    Obter a região ativa r da cabeça de Q
 3:    A partir de A[r].unplaced, determine os sucessores de r
 4:    para cada sucessor s de r faça
 5:       se A[s].examined = falso então
 6:          A[s].examined = verdadeiro
 7:          Computar A[s].unplaced, apagando o último vértice s de A[s].unplaced
 8:          se A[s].unplaced = ∅ então
 9:             retorne verdadeiro
10:          fim se
11:          Inserir s no final de Q
12:       fim se
13:    fim para
14: até Q = ∅
15: retorne falso
```

necessário observar que a condição na linha 5 junto com o laço da linha 4 implica que cada uma das regiões $A[s]$ será executada apenas uma vez, então as linhas de 6 a 11 serão executadas $O(n)$ vezes, com custo total em função do laço **repita-até** de $O(n^{k+1})$. Somando todas as contribuições, chega-se ao resultado do teorema 5.2.

Teorema 5.2. *Existe um algoritmo $O(n^{k+1})$ que responde à pergunta "$bw(G) \leq k$?" para um grafo G conexo de entrada, sendo k um inteiro positivo qualquer.*

Demonstração. Para verificar a viabilidade de aplicar o algoritmo 6 em G, executa-se uma busca em profundidade $O(m + n)$ no grafo, buscando por vértices com grau maior que $2k$. Caso existe um vértice desse tipo, sabe-se que a resposta à pergunta é **não**. Contudo, caso todos os vértices v de G tenham $d(v) \leq 2k$, então aplica-se o algoritmo 6 de custo $O(n^{k+1})$ em G. □

5.2 Algoritmo $O^*(5^n)$

Definição. *A notação $O^*(g(x))$ é usada para representar $O(h(n)g(x))$ para uma função polinomial $h(n)$, sendo n o tamanho da entrada do problema: $n \in \mathbb{N}$.*

Nesta seção será apresentado um algoritmo exato $O^*(5^n)$ para BANDWIDTH na classe geral de grafos que foi apresentado por Cygan e Pilipczuk, em 2008 [4]. Inicialmente, será tratado o caso $k \geq \frac{n}{3}$ e, posteriormente, o caso k geral. Os conceitos abordados na primeira análise, para $k \geq \frac{n}{3}$, serão fundamentais para estender para o caso geral.

5.2.1 Algoritmo para $k \geq \frac{n}{3}$

Algoritmo para $k \geq \frac{n}{2}$

Seja $G = (V, E)$ um grafo conexo e considere o problema BANDWIDTH-K (ver definição desse problema na seção 5.1) para o caso $k \geq \frac{n}{2}$. A ideia do algoritmo para resolver esse caso é particionar o conjunto V em dois conjuntos V_1 e V_2 com $|V_2| = k$, e verificar nessa partição se existe uma rotulação f de V tal que $bw_f(G) \leq k$. Como $|V_2| = k$ e $|V_1| = n-k \leq k$, as arestas a serem verificadas, para garantir $bw_f(G) \leq k$, devem ser apenas aquelas com um extremo em V_1 e o outro em V_2; arestas internas num dos conjuntos não precisam ser verificadas, em virtude do tamanho do conjunto.

Lema 5.1. *Seja $G = (V, E)$ um grafo com o conjunto de vértices V particionado em V_1 e V_2 tal que $|V_1| = s$ e $|V_2| = n - s$, sendo $s \leq k$, $n - s \leq k$ e $V_1 = \{v_1, \ldots, v_s\}$. Uma rotulação f de V_1 possui $bw_f(G) \leq k$ se, e somente se, para todo $1 \leq t \leq s$,*

$$\left| \bigcup_{i=1}^{t} N(v_i) \setminus V_1 \right| \leq t + k - s$$

Demonstração. ($\Rightarrow$) É fácil ver que a condição do teorema é necessária porque, sendo f rotulação de V_1 com $bw_f(G) \leq k$, então um vértice v_i de V_1 terá vértices adjacentes em V_2 com rótulos entre $s+1$ e $i+k$, logo vão existir, para v_i, no máximo $i + k - (s + 1) + 1 = k - s + i$ vértices em V_2. Como a inequação une a vizinhança $N(v_i) \setminus V_1$ para $1 \leq i \leq t$, o resultado do lema está correto.

($\Leftarrow$) Antes, considere $left(v)$, $v \in V_2$, o menor rótulo dos vértices adjacentes a v em V_1; se algum $v \in V_2$ não for adjacente a nenhum vértice em V_1, então

$left(v) = \infty$.

Para mostrar que a condição é suficiente, suponha que a equação do lema é válida para alguma permutação de V_1. Ordena-se os vértices v de V_2 em função da função $left(v)$. Na ordenação, se dois vértice de V_2 tiverem o mesmo valor de $left$, ordene-os em V_2 arbitrariamente.

Suponha por contradição que uma ordenação de V_2 de acordo com a função $left$ implique um vértice $v_x \in V_2$ com $left(v_x) = v_j$ tal que $j + k < x$. Pela ordenação de $left$, $\{v_{s+1}, \ldots, v_x\} \subset \cup_{i=1}^{j} N(v_i)/V_1$, e $| \cup_{i=1}^{j} N(v_i)/V_1| \geq x - s > j + k - s$: absurdo. □

Como consequência do lema 5.1, obtém-se o seguinte resultado, do lema 5.2.

Lema 5.2. *Seja G um grafo conforme o enunciado do lema 5.1. Existe um algoritmo $O^*(2^{n-k})$ que encontra uma rotulação f de G tal que $bw_f(G) \leq k$, atribuindo rótulos para V_1 e V_2 tal que $f(u) < f(v)$ para $u \in V_1$ e $v \in V_2$, se houver tal rotulação, ou afirma que tal rotulação não existe.*

Demonstração. Usa-se programação dinâmica sobre os subconjuntos de V_1 para verificar se uma rotulação f dos vértices de G satisfaz os critérios do lema 5.1; para cada subconjunto $A \subseteq V_1$, computa-se um valor $T(A)$ verdadeiro se, e somente se, os vértices de $A = (v_1, \ldots, v_{|A|})$ puderem ser ordenados tal que, para todo j, $1 \leq j \leq |A|$, $| \cup_{i=1}^{j} N(v_i) \setminus V_1| \leq j + k - (n - k)$. Essa verificação é equivalente a $T(A)$ é verdadeiro se, e somente se, $|N(A) \setminus V_1| \leq |A| + k - (n - k)$ e, para algum $v \in A$, $T(A \setminus \{v\})$ é verdadeiro. Com essa última informação pode-se aplicar programação dinâmica. Tal algoritmo consome $O^*(2^{n-k})$ em tempo. E a ordenação de V_2 pode ser encontrada com o uso da função $left$ definida no mesmo lema. □

Teorema 5.3. *Para $k \geq \frac{n}{2}$, pode-se encontrar uma rotulação f de G com $bw_f(G) \leq k$ em tempo $O^*(2^{2n-k})$.*

Demonstração. Como existem $\binom{n}{n-k} < 2^n$ opções de conjunto de V_1 com $n - k$ elementos, o algoritmo do lema 5.2 pode ser aplicado em cada uma delas, gerando uma complexidade final de $O^*(2^{2n-k})$. □

Algoritmo para $\frac{n}{3} \leq k < \frac{n}{2}$

A ideia neste caso é aplicar duas vezes o resultado do lema 5.1 numa partição de três conjuntos V_1, V_2 e V_3 de V para um grafo $G = (V, E)$, onde $|V_1| = s$,

$|V_2| = k$ e $|V_3| = n - k - s$ com $s = \lfloor \frac{n-k}{2} \rfloor \leq k$, e consequentemente $n - k - s \leq k$. Assim, apenas as arestas entre V_1 e V_2, e entre V_2 e V_3 são importantes numa verificação de que existe uma rotulação f de G tal que $bw_f(G) \leq k$. O algoritmo final gera todas as combinações de partições e verifica se existe uma rotulação f com $f(u) < f(v) < f(w)$ para $u \in V_1$, $v \in V_2$ e $w \in V_3$. Ao aplicar o lema 5.1 aos conjuntos $V_2 \cup V_3$ e $V_1 \cup V_2$, resulta o corolário 5.1.

Corolário 5.1. *Seja $G = (V, E)$ um grafo com o conjunto de vértices V particionado em V_1, V_2 e V_3 com $|V_1| = s$, $|V_2| = k$ e $|V_3| = n - s - k$, sendo $s = \lfloor \frac{n-k}{2} \rfloor \leq k$ e $n - s - k \leq k$. Uma rotulação f de $V_2 \cup V_3$ possui $bw_f(V_2 \cup V_3) \leq k$ se, e somente se, para todo $1 \leq t \leq k$,*

$$\left| \bigcup_{i=1}^{t} N(v_{s+i}) \cap V_3 \right| \leq t$$

O mesmo ocorre para $V_1 \cup V_2$ em que $1 \leq t \leq k$ com o seguinte resultado:

$$\left| \bigcup_{i=0}^{t-1} N(v_{s+k-i}) \cap V_1 \right| \leq t$$

O algoritmo final para este caso é obtido, como feito na seção anterior, usando programação dinâmica sobre todos os subconjuntos do conjunto V_2, calculando, para cada um desses subconjuntos, se as equações do corolário 5.1 são satisfeitas. Em função da rotulação encontrada para V_2, encontra-se o rótulo dos vértices de V_1 e V_3 pelo uso da função $left$, definida anteriormente. O custo da programação dinâmica para uma instância de V_2 numa partição de V é $O^*(2^k)$, e existem $\binom{n}{k}\binom{n-k}{s}$ partições de V, onde $\binom{n}{k}\binom{n-k}{s} \leq 4^n$, logo obtém-se o resultado do teorema 5.4.

Teorema 5.4. *Para $\frac{n}{3} \leq k < \frac{n}{2}$, pode-se encontrar uma rotulação f de G com $bw_f(G) \leq k$ em tempo $O^*(4^n)$.*

5.2.2 Algoritmo para $k \in \mathbb{N}$

Este caso geral, em que não há restrição sobre o valor de k, foi resolvido usando as ideias desenvolvidas nas subseções que trataram o caso $k \geq \frac{n}{3}$. Basicamente, a solução é encontrada ao particionar o conjunto V em conjuntos menores, e verificar os subconjuntos sobre a viabilidade de uma rotulação f que satisfaça $bw_f(G) \leq k$.

O algoritmo será executado em duas fases: uma para particionar o conjunto V de G, e outra para verificar se existe uma rotulação f de V baseada no particionamento que satisfaça $bw_f(G) \leq k$. Considere $G = (V, E)$ um grafo conexo.

Particionamento de V em segmentos de tamanho $k+1$ cada

Na primeira fase do algoritmo, o conjunto V será particionado em segmentos de tamanho $k+1$ cada; se $k+1 \nmid n$, então um segmento terá tamanho $n \mod k+1$. A divisão desta forma implica dois fatos importantes:

1. arestas internas num segmento não interessam na análise de '$bw_f(G) \leq k$?' para uma rotulação f, pois todas elas possuem diferença de rótulo nos extremos menor ou igual a k, e

2. qualquer aresta cujos extremos não estejam em segmentos consecutivos proíbe imediatamente uma rotulação f com $bw_f(G) \leq k$, obrigando o algoritmo a descartar o particionamento feito.

Apenas para esta primeira fase, seja $(v_1, \ldots, v_n)$ uma ordem qualquer dos vértices de G tal que também seja uma árvore geradora D no formato de raiz para folha (*root-to-leaf order*), no qual, se v_j é ascendente de v_i, então $j < i$; v_1 é a raiz da árvore.

Os segmentos são gerados como segue. Lembrar que, como os segmentos têm tamanho $k+1$, então existem no máximo $\lceil \frac{n}{k+1} \rceil$ possíveis segmentos distintos nos quais os vértices podem ser colocados.

1. Coloque a raiz, v_1, em qualquer um dos $\lceil \frac{n}{k+1} \rceil$ segmentos possíveis.

2. Para $i = 2, 3, \ldots, n$, faça:

   Seja v_j o acendente de v_i em D. Como $j < i$, o vértice v_j já foi colocado em algum segmento, então coloque v_i em um segmento distante no máximo uma unidade do segmento de v_j.

Tal colocação de vértices nos segmentos implica existir no máximo $n3^{n-1}$ atribuições, porque a raiz, inicialmente, pode ser colocada em qualquer segmento, e um filho pode estar em 3 segmentos diferentes em relação ao pai — no mesmo, à esquerda ou à direita.

Busca em profundidade sobre os segmentos construídos na primeira fase

Para cada atribuição dos vértices nos segmentos, na fase anterior, será gerada uma rotulação f de G tal que cada vértice de um segmento s ($1 \leq s \leq \lceil \frac{n}{k+1} \rceil$) tenha rótulo entre $(k+1)(s-1)+1$ e $(k+1)s$.

O importante nesta segunda fase é saber em que ordem se devem atribuir os rótulos aos vértices, para que se garanta a possibilidade de um algoritmo encontrar uma rotulação f com $bw_f(G) \leq k$ sempre que tal rotulação for possível. A ideia é fazer tal atribuição de forma que o algoritmo evolua, a cada passo, em estados consistentes (mais detalhes a seguir).

Antes, porém, é necessário verificar se a colocação dos vértices nos segmentos permite *a priori* que a rotulação f tenha $bw_f(G) \leq k$. Para isso, é preciso que o número de vértices alocados para cada segmento seja igual ao tamanho do segmento, $k+1$, e que também não existam arestas entre vértices de segmentos distantes mais de uma unidade. Se ambas essas condições são safisteitas, é possível continuar.

Uma observação importante é que, por causa das regras de atribuição da primeira fase do algoritmo, é de se esperar que não haja arestas entre segmentos distantes mais de um unidade, mas tais arestas podem existir. A distribuição não leva em consideração todas as arestas do grafo G mas apenas as de uma árvore geradora.

Neste momento, será demonstrado o lema 5.3 sobre a associação dos rótulos aos vértices, supondo que tal atribuição já tenha sido feita. A atribuição em si, porém, será apresentada mais tarde no algoritmo desta seção.

Definem-se $segment(f(v))$ e $color(f(v))$, para cada rótulo, como segue, sendo $f(v) \in \mathbb{N}$ o rótulo associado a um vértice v.

$$segment(f(v)) = \left\lceil \frac{f(v)}{k+1} \right\rceil$$
$$color(f(v)) = ((f(v)-1) \mod (k+1)) + 1$$

Lema 5.3. *A rotulação f possui $bw_f(G) \leq k$ se, e somente se, para toda aresta uv, se $segment(f(u)) < segmento(f(v))$, então $color(f(u)) > color(f(v))$.*

Demonstração. Como f respeita que vértices adjacentes sejam atribuídos a segmentos adjacentes, então $|segment(f(u)) - segmento(f(v))| \leq 1$. Se $segment(f(u)) = segmento(f(v))$, então $bw_f(G) \leq k$ porque um segmento tem tamanho máximo

$k+1$. Entretanto, se $segment(f(u)) < segmento(f(v))$ e $color(f(u)) = color(f(v))$, então a aresta uv tem tamanho $k+1$, logo $color(f(u)) > color(f(v))$, para que $bw_f(G) \leq k$. □

Corolário 5.2. *A rotulação f possui $bw_f(G) \leq k$ se, e somente se, para toda aresta uv com $segment(f(u))+1 = segmento(f(v))$ o vértice u é associado a uma posição maior que v na ordem de cor.*

Definição (Ordem de cor)**.** *Ordem de cor é a ordenação lexicográfica dos rótulos disponíveis de acordo com o par $(color(f(v)), segment(f(v)))$ para cada rótulo $f(v)$.*

A figura 5.1 exemplifica a ordem de cor de um grafo com 14 vértices e $k = 3$. A ordem de cor é usada para definir a ordem em que os rótulos serão atribuídos aos vértices; nesse exemplo, o rótulo 1 deve ser o primeiro a ser atribuído, seguido pelo rótulo 5, pelo 9, 13, 2, e assim por diante. Observa-se que o segundo rótulo a ser atribuído é o rótulo 5 em vez do rótulo 2.

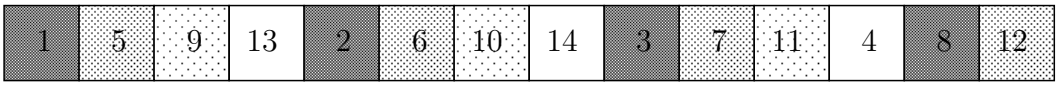

Figura 5.1: Exemplo de ordem de cor para os rótulos com $n = 14$ e $k = 3$.

Definição 1. *O estado do algoritmo será denotado por um conjunto de vértices A, $A \subseteq V$, satisfazendo:*

1. *vértices de A podem ser atribuídos às primeiras $|A|$ posições da ordem de cor, sendo compatíveis com a divisão dos vértices nos segmentos; e*
2. *não existe aresta uv com $u \in A$, $v \notin A$ e v atribuído a um segmento maior que o segmento atribuído a u.*

Lema 5.4. *Seja $\emptyset = A_0 \subset A_1 \subset \ldots \subset A_n = V$ uma sequência de estados com $v_i = A_i \setminus A_{i-1}$. Uma rotulação f que obedeça às regras da definição de estado, atribuindo o i-ésimo rótulo da ordem de cor a um vértice v_i, possui $bw_f(G) \leq k$.*

Demonstração. Suponha por contradição que f seja uma rotulação conforme o enunciado mas que $bw_f(G) > k$. Pelo corolário 5.2, existe uma aresta uv com $segment(f(u))+1 = segment(f(v))$ e $color(f(u)) \leq color(f(v))$. Então, u está antes de v na ordem de cor, que implica existir k tal que $u \in A_k$ e $v \notin A_k$. Entretanto, isso contradiz o fato de A_k ser um estado. □

Com os resultados apresentados até agora é possível descrever o algoritmo.

O algoritmo executa uma busca em profundidade sobre os estados a partir do estado $\emptyset$ (conjunto vazio) até V (todos os vértice do grafo), incluindo, a cada passo, um vértice do segmento do próximo rótulo a ser atribuído tal que o novo vértice incluído mantenha o algoritmo num estado consistente. É importante notar que o próximo rótulo a ser usado, segundo a ordem de cor, determina qual o segmento deve ser buscado para encontrar o novo vértice, e que, portanto, a vizinhança de busca é limitada aos $k+1$ vértices dentro do segmento, permitindo, assim, que o passo do algoritmo seja feito em tempo polinomial. O lema 5.4 garante que tal sequência seja encontrada se, e somente se, existir uma rotulação f tal que $bw_f(G) \leq G$. Apesar do resultado do algoritmo ser apenas se o *bandwidth* de um grafo G é (ou não) menor ou igual a um natural k, a rotulação dos vértices de G pode ser encontrada pela pilha da busca em profundidade executada.

O algoritmo acima executa em tempo $O^*(2^n)$ para cada atribuição de vértices aos segmentos. Como existem $n3^{n-1}$ atribuições diferentes, a complexidade final é $O^*(6^n)$. A próxima seção demonstra que a complexidade é $O^*(5^n)$.

Demonstrando complexidade $O^*(5^n)$ para o algoritmo da seção anterior

Teorema 5.5. *O algoritmo descrito na seção anterior, durante todas as atribuições de vértices a segmentos, visita no máximo $O(n5^{n-1})$ estados.*

Demonstração. Seja D a ordem raiz-para-folha dos vértices do grafo G usada no algoritmo, e sejam u e v dois vértices de D tal que u seja antecessor (pai) de v.

Se $u \notin A$, não é possível que $v \in A$ e $segment(v) < segment(u)$. Se $u \in A$, não é possível que $v \in A$ e $segment(v) > segment(u)$. Essas restrições quanto à evolução dos estados do algoritmo são consequencias do corolário 5.2. Assim, sendo que a raiz não possui tais restrições, todo estado pode ser estendido com um vértice apenas por cinco possibilidades. Conclui-se que existem no máximo $O(n5^{n-1})$ estados possíveis para a evolução do algoritmo. □

Capítulo 6

Conclusão

O objetivo deste trabalho foi compilar os principais resultados sobre a solução por algoritmos exatos para o problema de *bandwidth* de grafos e sobre a demonstração de que o problema é NP-completo para grafos em geral. Este problema de otimização combinatória é importante por ter várias aplicações mas ser de dificuldade amplamente reconhecida.

No decorrer do trabalho foram apresentados com notação e abordagem uniforme os algoritmos polinomiais mais conhecidos para classes específicas de grafos. Outro aspecto abordado foram os algoritmos exponenciais exatos que resolvem o problema para grafos em geral. Foram ainda apresentadas duas demonstrações de que o problema é NP-completo.

O capítulo de descrição formal do problema inclui ainda equações que estabelecem o *bandwidth* para algumas classes de grafos simples, tais como ciclos, caminhos, e bipartidos completos. Também nesse capítulo, foram incluídos resultados sobre limites inferiores usados nas provas dos algoritmos polinomiais apresentados ao longo do texto. O capítulo termina com um estudo sobre as relações entre os conceitos de *bandwidth* e *pathwidth*, sendo este último outro problema associado a largura de grafos, também de reconhecida dificuldade.

Entre os algoritmos polinomiais apresentados estão os algoritmos em tempo linear para as classes *quasi-threshold*, cografos e P_4-esparso, com relação de inclusão, nessa ordem, entre elas. Devido às semelhanças estruturais dessas com as classes P_4-*lite* e P_4-*tidy* [8], que incluem, nessa ordem, as classes anteriores, tem-se um assunto para investigação futura, mais especificamente tentar adaptar os algoritmos conhecidos, para resolver o problema eficientemente nas superclasses.

Um outro assunto para investigação futura é a busca de algoritmos exatos para a classe de grafos de permutação — grafos de interseção de segmentos de reta entre duas retas paralelas —, na qual a solução do problema BANDWIDTH é frequentemente citado na literatura como sendo de grande interesse.

Referências Bibliográficas

[1] S. F. Assmann, G. W. Peck, M. M. Syslo, and J. Zak. The bandwidth of caterpillars with hairs of length 1 and 2. *SIAM Journal on Algebraic and Discrete Methods*, 2:387–393, 1981.

[2] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[3] D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14:926–934, 1985.

[4] M. Cygan and M. Pilipczuk. Faster exact bandwidth. *Lecture Notes in Computer Science*, 3544:101–109, 2008.

[5] J. Díaz, M. D. Penrose, J. Petit, and M. J. Serna. Linear orderings of random geometric graphs. *Lecture Notes in Computer Science*, 1665:291–302, 1999.

[6] M. R. Garey, R. L. Graham, D. S. Johnson, and D. E. Knuth. Complexity results for bandwidth minimization. *SIAM Journal on Applied Mathematics*, 34:477–495, 1978.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[8] V. Giakoumakis, F. Roussel, and H. Thuillier. On P_4-tidy graphs. *Discrete Mathematics and Theoretical Science*, 1:17–41, 1997.

[9] N. E. Gibbs, W. G. Poole J., and P. K. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis*, 13:236–250, 1976.

[10] M. C. Golumbic. Trivially perfect graphs. *Discrete Mathematics*, 24:105–107, 1978.

[11] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[12] E. Gurari and I. H. Sudborough. Improved dynamic programming algorithms for the bandwidth minimization and the mincut linear arrangement problem. *Journal of Algorithms*, 5:531–546, 1984.

[13] P. Heggernes, D. Kratsch, and D. Meister. Bandwidth of bipartite permutation graphs in polynomial time. *Journal of Discrete Algorithms*, 7:533–544, 2009.

[14] L. T. Q. Hung, M. M. Syslo, M. L. Weaver, and D. B. West. Bandwidth and density for block graphs. *Discrete Mathematics*, 189:163–176, 1998.

[15] B. Jamison and S. Olariu. Recognizing P_4-sparses graphs in linear time. *SIAM Journal on Computing*, 21:381–406, 1992.

[16] B. Jamison and S. Olariu. A tree representation for P_4-sparses graphs. *Discrete Applied Mathematics*, 35:115–129, 1992.

[17] H. Kaplan and R. Shamir. Pathwidth, bandwidth and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing*, 25:540–561, 1996.

[18] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[19] D. J. Kleitman and R. V. Vohra. Computing the bandwidth of interval graphs. *SIAM Journal on Discrete Mathematics*, 3:373–375, 1990.

[20] T. Kloks, D. Kratsch, Y. L. Borgne, and H. Müller. Bandwidth of split and circular permutation graphs. In *WG '00: Proceedings of the 26th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 243–254, London, UK, 2000. Springer-Verlag.

[21] T. Kloks, D. Kratsch, and H. Müller. Bandwidth of chain graphs. *Information Processing Letters*, 68:313–315, 1998.

[22] T. Kloks, D. Kratsch, and H. Müller. Approximating the bandwidth for AT-free graphs. *Journal of Algorithms*, 32:41–57, 1999.

[23] D. Kratsch. Finding the minimum bandwidth of an interval graph. *Information and Computation*, 74:140–158, 1987.

[24] D. Kratsch and L. Stewart. Approximating bandwidth by mixing layouts of interval graphs. *SIAM Journal on Discrete Mathematics*, 15:435–449, 2002.

[25] Y. Lai and K. Williams. A survey of solved problems and applications on bandwidth, edgesum, and profile of graphs. *Journal of Graph Theory*, 31:75–94, 1999.

[26] S. Ma, W. D. Wallis, and J. Wu. Optimization problems on quasi-threshold graphs. *Journal of Combinatorics, Information and System Sciences*, 14:105–110, 1989.

[27] R. Mahesh, C. P. Rangan, and A. Srinivasan. On finding the minimum bandwidth of interval graphs. *Information and Computation*, 95:218–224, 1991.

[28] B. Monien. The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 7:505–512, 1986.

[29] C. H. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.

[30] G. W. Peck and A. Shastri. Bandwidth of theta graphs with short paths. *Discrete Mathematics*, 103:177–187, 1992.

[31] J. B. Saxe. Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM Journal on Algebraic and Discrete Methods*, 1:363–369, 1980.

[32] J. P. Spinrad. *Efficient Graph Representations*, volume 19. Fields Institute Monographs, 2003.

[33] A. P. Sprague. An $O(n \log n)$ algorithm for bandwidth of interval graphs. *SIAM Journal on Discrete Mathematics*, 7:213–220, 1994.

[34] E. S. Wolk. The comparability graph of a tree. *Proceedings of the American Mathematical Society*, 3:789–795, 1962.

[35] J. H. Yan. The bandwidth problem in cographs. *Tamsui Oxford Journal of Mathematical Sciences*, 13:31–36, 1997.

[36] J. H. Yan. Algorithm aspects of the bandwidth problem on P_4-sparse graphs. *Tamsui Oxford Journal of Mathematical Sciences*, 14:11–18, 1998.

[37] J. H. Yan, J. J. Chen, and G. J. Chang. Quasi-threshold graphs. *Discrete Applied Mathematics*, 69:247–255, 1996.