UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

CAROLINA SZKRUC DE CARVALHO

ALGORITMOS POLINOMIAIS PARA PROBLEMAS DE LAYOUT EM GRAFOS

Rio de Janeiro
2015

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

CAROLINA SZKRUC DE CARVALHO

ALGORITMOS POLINOMIAIS PARA PROBLEMAS DE LAYOUT EM GRAFOS

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Mestre em Informática.

Orientador: Lilian Markenzon (D.Sc., UFRJ)
Co-orientador: Luziane Ferreira de Mendonça (D.Sc., UFRJ)

Rio de Janeiro
2015

C331 Carvalho, Carolina Szkruc de

Algoritmos Polinomiais para Problemas de Layout em Grafos / Carolina Szkruc de Carvalho. – 2015.
75 f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática, Rio de Janeiro, 2015.

Orientador: Lilian Markenzon (D.Sc., UFRJ).
Co-orientador: Luziane Ferreira de Mendonça (D.Sc., UFRJ).

1. Grafos caterpillar. 2. Separação de vértices. 3. Arranjo linear mínimo. 4. Largura de banda. 5. Perfil. 6. Problemas de layout. – Teses. I. Markenzon (D.Sc., UFRJ), Lilian (Orient.). II. Mendonça (D.Sc., UFRJ), Luziane Ferreira de (Co-orient.). III. Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática. IV. Título

CDD

CAROLINA SZKRUC DE CARVALHO

Algoritmos Polinomiais para Problemas de Layout em Grafos

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Mestre em Informática.

Aprovado em: Rio de Janeiro, _____ de _____ de _____.

_____

Prof. Dra. Lilian Markenzon (D.Sc., UFRJ) (Orientador)

_____

Prof. Dra. Luziane Ferreira de Mendonça (D.Sc., UFRJ) (Co-orientador)

_____

Prof. Claudia Marcela Justel (D.Sc., IME)

_____

Prof. Nair Maria Maia de Abreu (D.Sc., UFRJ)

_____

Prof. Vinícius Gusmão P. de Sá (D.Sc., UFRJ)

Rio de Janeiro
2015

Para Daniel, David e Helena
com todo amor e gratidão.

AGRADECIMENTOS

Agradeço aos meus pais, que dedicaram suas vidas para que eu e minha irmã tivéssemos uma educação de qualidade. A minha irmã pela paciência nos meus momentos de estresse desde a época da graduação. Em especial agradeço ao companheiro e amor da minha vida Daniel, que me apoiou em todos os momentos: nas provas, na qualificação, no artigo e em todos os outros momentos. Sempre me dando força e me acalmando, sem ele e o meu pai eu não teria o suporte necessário para cursar o mestrado. Também sou grata a todo Corpo Docente do Departamento de Ciência da Computação e do Programa de Pós-Graduação em Informática. Aos amigos que fiz durante esse período, em especial a Anne Federici que foi uma super companheira de jornada, compartilhamos os momentos de aflição e passamos as madrugadas finais apoiando uma a outra. Agradeço as minhas orientadoras Lilian Markenzon e Luziane F. de Mendonça, pela paciência, dedicação e preocupação. Elas foram mais que orientadoras foram mestres e amigas, passando conhecimento e experiências. Agradeço a Deus, que permitiu que eu nascesse na minha família e que colocou todas essas pessoas no meu caminho pela terra, que me ajudaram nessa conquista. Finalmente agradeço a CAPES pelo suporte financeiro.

RESUMO

Carvalho, Carolina Szkruc de. **Algoritmos Polinomiais para Problemas de Layout em Grafos**. 2015. 72 f. Dissertação (Mestrado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2015.

Problemas de layout de grafos são problemas de otimização combinatória cujo objetivo é encontrar um layout linear de um grafo de tal forma que um certo custo seja otimizado. São conhecidos na literatura resultados eficientes para esses problemas em algumas classes de grafos.

Neste trabalho consideramos uma importante subclasse de árvores, os grafos caterpillar, e apresentamos soluções ótimas de complexidade linear para quatro problemas de layout em grafos: separação de vértices, arranjo linear mínimo, largura de banda e perfil. Mostramos que o valor da separação de vértices para essa classe é constante. Também provamos o valor do arranjo linear mínimo para estrelas e caminhos, que são tipos especiais de caterpillars; além disso, foi mostrado como montar o layout ótimo desse problema para caterpillars. Para o problema de largura de banda a solução foi dada para um tipo especial de caterpillar, que denominamos caterpillar 2-estrelas. O último problema estudado foi o perfil: foi visto que o valor dele para um caterpillar está diretamente ligado ao número de vértices do grafo.

Palavras-chave: Grafos caterpillar, Separação de vértices, Arranjo linear mínimo, Largura de banda, Perfil, Problemas de layout.

ABSTRACT

Carvalho, Carolina Szkruc de. **Algoritmos Polinomiais para Problemas de Layout em Grafos**. 2015. 72 f. Dissertação (Mestrado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2015.

Graph layout problems are combinatorial optimization problems whose goal is to find a linear layout for a graph in such way that the cost is optimized. For these problems, efficient solutions for some classes of graphs have been known in the literature.

In this work we consider an important subclass of trees, the caterpillars, and we present optimal solutions with linear time complexity for four problems: the vertex separation problem, minimum linear arrangement problem, bandwidth problem and the profile problem. We show that the value of the vertex separation to this class is constant. We also proved the value of the minimum arrangement linear for stars and paths, which are special types of caterpillars; furthermore, it was shown how to assemble the optimal layout of this problem for caterpillars. For the bandwidth problem, the solution was given to a special type of caterpillar, called caterpillar-2 stars. For the profilo problem it was seen that his value to a caterpillar is directly linked to the number of vertices of the graph.

Keywords: Vertex separation, Minimum linear arrangement, Bandwidth, Profile, Caterpillar.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Problemas de layout em grafos são problemas de otimização combinatória cujo objetivo é encontrar um layout linear de um grafo G dado de tal maneira que uma determinada função objetivo seja otimizada. Um *layout linear* de um grafo $G = (V, E)$, com $|V| = n$ vértices, é uma função bijetiva $L : V \to [n] = \{1, 2, 3, ..., n\}$, isto é, um layout é uma rotulação dos vértices do grafo em inteiros distintos.

Um abrangente levantamento do assunto foi feito por Díaz *et al.* em [12] e por Petit em [42], onde pode ser visto que a solução algorítmica de grande parte desses problemas é exponencial. Na pesquisa por soluções polinomiais é usual examinar classes conhecidas, visando reconhecer propriedades estruturais particulares das classes que permitam encontrar essa solução. Dentre essas classes podemos citar os grafos completos, as árvores e os grafos planares, entre outros.

O objetivo desta dissertação é estudar alguns problemas de layouts em grafos, apresentando soluções polinomiais encontradas para algumas classes. Pretende-se também buscar casos particulares em que a solução já conhecida possa ser significativamente simplificada. Os problemas estudados serão: *separação de vértices, arranjo linear mínimo, largura de banda e perfil.* Os quatro problemas são NP-completos para grafos gerais [12]. Iniciamos o estudo verificando as soluções polinomiais existentes para árvores. Em seguida desenvolvemos soluções polinomiais para a classe dos caterpillars, que é uma subclasse especial de árvores. Um *caterpillar* é uma árvore T, com pelo pelo menos 3 vértices, que verifica a seguinte condição: quando eliminamos todas as folhas de T obtemos um caminho de comprimento não negativo [14].

O primeiro capítulo apresenta alguns conceitos básicos importantes. No Capítulo 2 será feita uma apresentação dos problemas estudados. Nos Capítulos 3 a 6 trataremos dos problemas: separação de vértices, arranjo linear mínimo, largura de banda e perfil. Em cada um dos capítulos, respectivamente, é introduzida a solução particular para os grafos caterpillars. No Capítulo 7 resumimos os resultados.

1.1 Conceitos Básicos

Nesta seção revisaremos alguns conceitos básicos de grafos pertinentes à dissertação. Assume-se a familiaridade com os conceitos básicos de grafos que podem ser encontrados em Diestel [14].

Um grafo G é um par $G = (V, E)$, onde V é um conjunto de vértices e E é um conjunto de arestas, sendo cada aresta um subconjunto de V com 1 ou 2 vértices. Quando esse subconjunto possui apenas um vértice essa aresta é denominada *laço*. Nesta dissertação consideraremos grafos simples, ou seja, sem laços. Para denominar o número de vértices e arestas denotamos $n = |V|$ e $m = |E|$. Uma aresta será denotada por uv.

Dois vértices são *adjacentes* ou *vizinhos* quando são extremidades de uma aresta. O *conjunto de adjacência* é o conjunto que contém todos os vizinhos de um determinado vértice $v \in V$. Denota-se $N(u) = \{v \in V : uv \in E\}$ o conjunto de adjacência do vértice u e $N[u] = \{u\} \cup N(u)$. O número de elementos do conjunto de adjacência de um determinado vértice v é denominado *grau* de v, e é denotado por $d(v)$.

Um grafo $G' = (V', E')$ é um *subgrafo* quando $V' \subseteq V$ e $E' \subseteq E$. *Subgrafo induzido* $G[S]$ é um subgrafo de G tal que $S \subseteq V$ e possui todas as arestas de E

com extremidades em S.

Um *caminho simples* é uma sequência $< v_0, v_1, ..., v_p >$, $p \geq 0$, de vértices distintos tal que $v_i v_{i+1} \in E$ para $0 \leq i \leq p$. O *comprimento* do caminho é o número de arestas que o constitui. Um grafo é *conexo* quando existir pelo menos um caminho entre cada par de vértices. A *distância* de um vértice v a um vértice u é o menor comprimento de caminho que existe de v até u. O *diâmetro* de G é a maior distância existente entre dois vértices de G, em outras palavras é o maior caminho existente em G. Denotamos diâmetro de G por $diam(G)$.

Uma *árvore* é um grafo conexo tal que $m = n - 1$. Uma *árvore enraizada* T, é uma árvore na qual algum vértice é escolhido como especial, a raíz da árvore. Seja $T = (V, E)$ uma árvore enraizada e $v \in V$ um vértice de T. O *nível* de v em T é o comprimento do caminho da raiz até o vértice v. A *altura* de uma árvore é igual ao valor máximo de nível dentre todos os seus vértices. Uma subárvore de T é uma árvore enraizada em algum vértice de T.

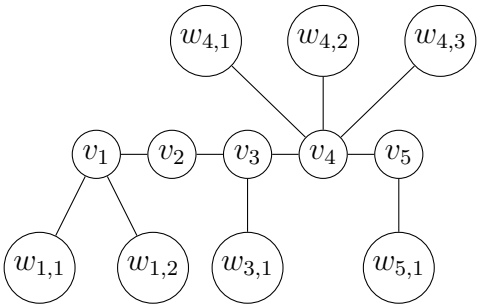

Figura 1.1: Exemplo de um caterpillar G.

Um *grafo caterpillar* (ou simplesmente caterpillar) é uma árvore T, com pelo menos 3 vértices, que verifica a seguinte condição: quando eliminamos todas as folhas de T obtemos um caminho de comprimento não negativo. O caminho assim obtido no caterpillar é chamado de *espinha dorsal* do caterpillar. Vamos denotar por v_i, $i = 1, \ldots, n_1$, os vértices que fazem parte do caminho e por $w_{i,j}$ os vértices

folhas, onde i indica o vértice v_i da espinha dorsal ao qual a folha é adjacente, j variando de 1 até o número de folhas adjacentes a v_i. O número de vértices do caterpillar é então $n = n_1 + n_2$, onde n_2 é a sua quantidade total de folhas. A Figura 1.1 ilustra a notação utilizada.

Nesta dissertação serão considerados grafos conexos.

2 PROBLEMAS

2.1 Introdução

Como já foi dito, um *layout linear* é uma rotulação dos vértices de um grafo com inteiros sequenciais e distintos e foi também chamado na literatura de *ordem linear* [1] ou uma *numeração* [8]. Dois trabalhos com levantamentos cuidadosos dos resultados existentes na área são os *surveys* de [12] e [42].

Neste capítulo será apresentado uma introdução dos problemas estudados nessa dissertação. Antes de apresentar os problemas tratados alguns conceitos serão apresentados:

Dado um grafo $G = (V, E)$, $\mathcal{L} = \{L_1, L_2, L_3, ..., L_l\}$ denota o conjunto de todos os layouts de G. Seja L um layout de um grafo G e i um inteiro fixo em $[n]$; definimos os conjuntos $\mathcal{E}(i, L, G)= \{u \in V : L(u) \leq i\}$ e $\mathcal{D}(i, L, G)= \{u \in V : L(u) > i\}$ como, respectivamente, os vértices à esquerda e à direita do vértice w mapeado em i (ou seja, $L(w) = i$). Dado um grafo G, a *separação de vértices* na posição i é definida por $\delta(i, L, G) = |\{u \in \mathcal{E}(i, L, G) : \exists v \in \mathcal{D}(i, L, G) \text{ com } uv \in E\}|$. Um *layout* L é representado por uma sequência de pares constituídos por vértices do grafo e suas posições em L :

$$L = \{(v_1, L(v_1)), (v_2, L(v_2)), ..., (v_n, L(v_n))\}.$$

O *comprimento de aresta* no layout L é definido por $\lambda(uv, L, G) = |L(u) - L(v)|$, onde de $uv \in E$. O custo do layout é uma função F que associa cada layout L de G a um inteiro $F(L, G)$. Seja F o custo de layout. O problema de otimização de um layout consiste em determinar o L^* de G tal que

$$F(L^*, G) = \min_{L \in \mathcal{L}} F(L, G)$$

sendo $\mathcal{L}$ o conjunto de todos os layouts de G.

Seja G um grafo (Figura 2.1) e sejam L_1, L_2, L_3 e L_4 alguns de seus layouts (Figura 2.2).

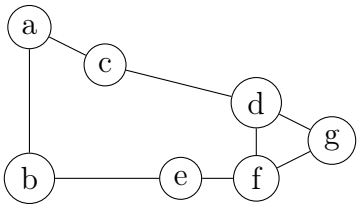

Figura 2.1: Grafo G.

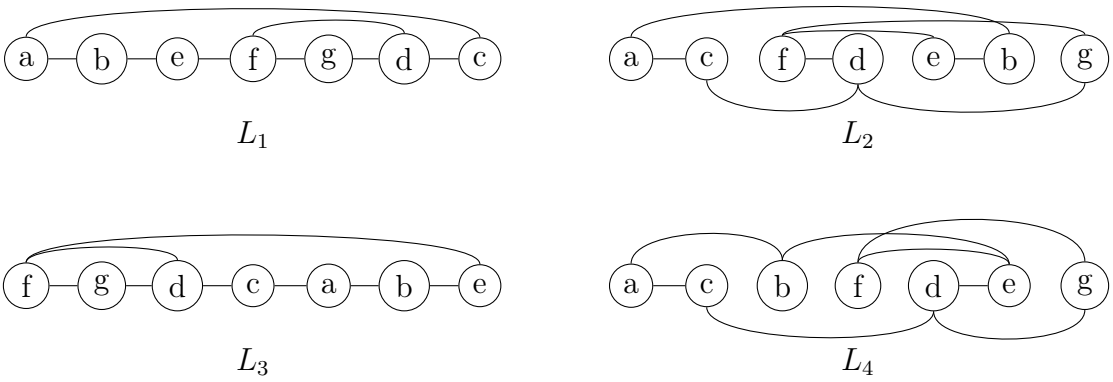

Figura 2.2: Layouts de G.

2.2 Problema 1: Separação de Vértices

O problema da separação de vértices possui aplicações em algoritmos para VLSI design [34]. Este problema possui relação com alguns outros problemas bem conhecidos, essas relações serão apresentadas no Capítulo 3.

A separação de vértices de um layout, $vs(L, G)$, é a maior separação de vértices, $\delta(i, L, G)$, dentre todos os vértices do layout L.

Seja $vs(L, G) = \max_{i \in [n]} \delta(i, L, G)$, onde $[n] = \{1, 2, ..., n\}$. Dado um grafo $G = (V, E)$ o problema da *determinação da separação de vértices* consiste em encontrar um layout L^* tal que

$$vs(L^*, G) = \min_{L \in \mathcal{L}} vs(L, G),$$

sendo $\mathcal{L}$ o conjunto de todos os layouts de G. O valor $vs(L^*, G)$ é denotado MINVS(G).

Seja G um grafo (Figura 2.1) e sejam L_1, L_2, L_3 e L_4 quatro de seus layouts (Figura 2.2). Um dos layouts ótimos do grafo G é o L_3. Segue abaixo a separação de vértices para cada um dos quatro layouts apresentados na Figura 2.2:

- $vs(L_1, G) = \max_{i \in [n]} \delta(i, L_1, G) = 3$

  $\delta(1, L_1, G) = 1$

  $\delta(2, L_1, G) = 2$

  $\delta(3, L_1, G) = 2$

  $\delta(4, L_1, G) = 2$

  $\delta(5, L_1, G) = 3$

  $\delta(6, L_1, G) = 2$

  $\delta(7, L_1, G) = 0$

- $vs(L_2, G) = \max_{i \in [n]} \delta(i, L_2, G) = 4$

$\delta(1, L_2, G) = 1$

$\delta(2, L_2, G) = 2$

$\delta(3, L_2, G) = 3$

$\delta(4, L_2, G) = 3$

$\delta(5, L_2, G) = 4$

$\delta(6, L_2, G) = 2$

$\delta(7, L_2, G) = 0$

- $vs(L_3, G) = \max_{i \in [n]} \delta(i, L_3, G) = 2$

  $\delta(1, L_3, G) = 1$

  $\delta(2, L_3, G) = 2$

  $\delta(3, L_3, G) = 2$

  $\delta(4, L_3, G) = 2$

  $\delta(5, L_3, G) = 2$

  $\delta(6, L_3, G) = 2$

  $\delta(7, L_3, G) = 0$

- $vs(L_4, G) = \max_{i \in [n]} \delta(i, L_4, G) = 3$

  $\delta(1, L_4, G) = 1$

  $\delta(2, L_4, G) = 2$

  $\delta(3, L_4, G) = 2$

  $\delta(4, L_4, G) = 3$

  $\delta(5, L_4, G) = 3$

  $\delta(6, L_4, G) = 2$

  $\delta(7, L_4, G) = 0$

2.3 Problema 2: Arranjo Linear Mínimo

O arranjo linear mínimo foi considerado por [37] como um modelo para algumas atividades do sistema nervoso. Outra aplicação é no escalonamento de tarefas em uma única máquina (single machine job scheduling) [1, 43].

O arranjo linear mínimo é a soma de todos os comprimentos de aresta de um determinado layout.

Seja $LA(L,G) = \sum_{uv \in E} \lambda(uv, L, G)$. Dado $G = (V, E)$ o problema da *determinação do arranjo linear mínimo* consiste em encontrar um layout L^* tal que

$$LA(L^*, G) = \min_{L \in \mathcal{L}} LA(L, G),$$

sendo $\mathcal{L}$ o conjunto de todos os layouts de G. O valor $LA(L^*, G)$ é denotado $\text{MINLA}(G)$.

Seja G um grafo (Figura 2.1) e sejam L_1, L_2, L_3 e L_4 quatro de seus layouts (Figura 2.2). Temos que dois dos layouts ótimos do grafo G são L_1 e L_3. Segue abaixo o arranjo linear mínimo para cada um dos quatro layouts apresentados na Figura 2.2:

- $LA(L_1, G) = \sum_{uv \in E} \lambda(uv, L_1, G) = 14$

  $6 + 1 + 1 + 1 + 2 + 1 + 1 + 1 = 14$

- $LA(L_2, G) = \sum_{uv \in E} \lambda(uv, L_2, G) = 19$

$5+1+2+2+4+1+3+1=19$

- $LA(L_3,G)=\sum_{uv\in E}\lambda(uv,L_3,G)=14$

  $6+2+1+1+1+1+1+1=14$

- $LA(L_4,G)=\sum_{uv\in E}\lambda(uv,L_4,G)=17$

  $2+1+3+3+2+3+1+2=17$

2.4 Problema 3: Largura de Banda

A largura de banda de um layout L é o maior comprimento de aresta desse layout. Dentre as várias aplicações deste problema, destacam-se: solução de sistemas lineares e solução de problemas de interconexão de redes [33].

Seja $BW(L,G)=\max_{uv\in E}\lambda(uv,L,G)$. Dado um grafo $G=(V,E)$ o problema da *determinação da largura de banda* consiste em encontrar um layout L^* tal que

$$BW(L^*,G)=\min_{L\in\mathcal{L}}BW(L.G),$$

sendo $\mathcal{L}$ o conjunto de todos os layouts de G. O valor $BW(L^*,G)$ é denotado $\textsc{minbw}(G)$.

Seja G um grafo (Figura 2.1) e sejam L_1, L_2, L_3 e L_4 quatro de seus layouts (Figura 2.2). Temos que um dos layouts ótimos do grafo G é o L_4. Segue abaixo a largura de banda para cada um dos quatro layouts apresentados na Figura 2.2:

- $BW(L_1, G) = \max_{uv \in E} \lambda(uv, L_1, G) = 6$
- $BW(L_2, G) = \max_{uv \in E} \lambda(uv, L_2, G) = 5$
- $BW(L_3, G) = \max_{uv \in E} \lambda(uv, L_3, G) = 6$
- $BW(L_4, G) = \max_{uv \in E} \lambda(uv, L_4, G) = 3$

2.5 Problema 4: Perfil

O problema do perfil é muito usado para melhorar o desempenho das operações em sistemas de equações não lineares [46]; ele também possui aplicações no projeto Genoma Humano [25], em arqueologia [26], na recuperação da informação [7] e reconhecimento de impressões digitais [25]. O perfil é a diferença absoluta dos rótulos dos vértices com seus adjacentes de menor valor.

Seja $PR(L, G) = \sum_{u \in V} \left(L(u) - \min_{v \in N[u]} L(v) \right)$. Dado um grafo $G = (V, E)$, o problema de *determinação do perfil* de G consiste em encontrar um layout L^* tal que

$$PR(L^*, G) = \min_{L \in \mathcal{L}} PR(L, G),$$

sendo $\mathcal{L}$ o conjunto de todos os layouts de G. O valor $PR(L^*, G)$ é denotado $\text{MINPR}(G)$.

Seja G um grafo (Figura 2.1) e sejam L_1, L_2, L_3 e L_4 quatro de seus layouts (Figura 2.2). Alguns dos layouts ótimos do grafo G são L_1, L_3 e L_4. Segue abaixo o perfil para cada um dos quatro layouts apresentados na Figura 2.2:

- $PR(L_1, G) = \sum_{u \in V} \left(L(u) - \min_{v \in N[u]} L(v) \right) = 12$

  $0 + 1 + 1 + 1 + 1 + 2 + 6 = 12$

- $PR(L_2, G) = \sum_{u \in V} \left(L(u) - \min_{v \in N[u]} L(v) \right) = 14$

  $0 + 1 + 0 + 2 + 2 + 5 + 4 = 14$

- $PR(L_3, G) = \sum_{u \in V} \left(L(u) - \min_{v \in N[u]} L(v) \right) = 12$

  $0 + 1 + 2 + 1 + 1 + 1 + 6 = 12$

- $PR(L_4, G) = \sum_{u \in V} \left(L(u) - \min_{v \in N[u]} L(v) \right) = 12$

  $0 + 1 + 2 + 0 + 3 + 3 + 3 = 12$

Nesse capítulo foram apresentados os problemas estudados nesta dissertação. O primeiro problema tratado, no próximo capítulo, será a *separação de vértices.*

3 SEPARAÇÃO DE VÉRTICES

3.1 Introdução

O problema da separação de vértices foi introduzido por Lengauer (1981), que definiu o que chamou um *jogo de separação de vértices*. Consideramos aqui o mesmo conceito, em termos de layouts lineares. A versão de decisão do problema de separação de vértices para grafos em geral foi provada no mesmo trabalho ser um problema NP-completo.

Seja $vs(L, G) = \max_{i \in [n]} \delta(i, L, G)$, onde $[n] = \{1, 2, ..., n\}$. Dado um grafo $G = (V, E)$ o problema da *determinação da separação de vértices* consiste em encontrar um layout L^* tal que

$$vs(L^*, G) = \min_{L \in \mathcal{L}} vs(L, G),$$

sendo $\mathcal{L}$ o conjunto de todos os layouts de G. O valor $vs(L^*, G)$ é denotado MINVS(G).

O problema da separação de vértices ganhou importância devido a sua relação, direta ou indireta, com outros problemas importantes de grafos, tais como número de busca (search number - também chamado de número de aresta de busca (edge search number) [38]), espessura do intervalo (interval thickness)[27] e largura de caminho (pathwidth) [6].

O conceito de número de busca foi introduzido por Parsons [39, 40]. O *número*

de busca de um grafo G, denotado por $s(G)$, é o número mínimo de buscadores necessários para garantir a captura de um fugitivo que pode mover-se com velocidade arbitrária sobre as arestas do grafo. Esses buscadores desenvolvem um chamado *plano de busca*. Um plano de busca permite três tipos de operações: colocar um buscador em um vértice, remover um buscador de um vértice, movimentar o buscador ao longo de uma aresta. Um plano de busca que utiliza o menor número de buscadores é um plano de busca ótimo [35]. Foi mostrado por [15] uma relação entre separação de vértices e número de busca: a separação de vértices de um grafo G é menor ou igual ao número de busca de G, e o número de busca de G será menor ou igual a separção de vértices de G mais um. Em [15] támbem é dado uma transformação de um grafo G em G' tal que a separação de vértices de G' é igual ao número de busca de G.

Em 1986 Kirousis and Papadimitriou definiram uma variação do número de busca, chamado número de nó de busca [28]. A diferença entre o número de busca e o número de nó de busca é que no segundo o buscador não se move ao longo da aresta. Foi mostrado que o número de nó de busca é igual a separação de vértices mais um.

Conforme mencionado, outro problema que possui grande relação com a separação de vértices é o problema de determinar espessura do intervalo, θ, que foi introduzido em 1985 por Kirousis e Papadimitriou [27]. Dado um conjunto de intervalos I numa reta, o grafo de intervalos de I é obtido representando cada intervalo por um vértice e cada interseção de dois intervalos por uma aresta entre os vértices correspondentes. A espessura de intervalo de um grafo é a menor clique máxima obtida dentre todos os grafos que são supergrafos de G [27].

O problema de determinar a largura de um caminho de um grafo G, $pw(G)$, foi introduzido por [44] em 1983. Uma *decomposição em caminho* de um grafo é

uma sequência de subconjuntos de vértices, $D = X_1, ...X_r$, onde cada aresta do grafo G possui as duas extremidades em algum conjunto X_i; se um vértice do grafo G pertence aos conjuntos X_i e X_j onde $i < j$, então o vértice deve pertencer a todos os conjuntos X_k onde $i < k < j$. A largura de caminho do grafo G com respeito a D é o maior número de vértices nos conjuntos X_i , $1 \leq i \leq r$, menos 1 unidade. A largura de caminho de G é definida como a menor largura de G considerando todas as possíveis decomposições em caminho de G [44], [45].

parâmetro	**relação com a separação de vértices**
número de busca $s(G)$	$vs(G) \leq s(G) \leq vs(G) + 1$
espessura de intervalo $\theta(G)$	$\theta(G) = vs(G) + 1$
largura de caminho $pw(G)$	$pw(G) = vs(G)$

A determinação da separação de vértices tem solução polinomial para algumas classes de grafos. Díaz [12] e Petit [42] listam alguns desses resultados. Bodlaender e Mohring [6] mostraram um algoritmo com complexidade de tempo de $O(n)$ que determina a largura de caminho para cografos. Para separação de vértices dos grafos grade com n dimensões existem resultados com complexidade de tempo de $O(n^2)$ [5].

Ellis *et al.* [15] apresentaram um algoritmo que computa a separação de vértices e o layout ótimo para árvores com complexidade de tempo $O(n \log n)$. Ainda para árvores, Skodinis [50] estabelece um algoritmo com tempo de complexidade de $O(n)$. Ellis e Markov [16] apresentaram um algoritmo, com complexidade de tempo de $O(n \log n)$, que determina a separação de vértices e o layout ótimo para grafos uniciclos (grafo composto por uma árvore acrescida de uma aresta). São usados no algoritmo os métodos descritos em [15] e [50] para árvores. Algum tempo depois, Chou *et al.* [9] modificaram o algoritmo dado por [16] chegando assim em uma complexidade de tempo de $O(n)$. Kawasaki e Justel [38] mostraram que a separação de vértices de uma árvore binária cheia T_h de altura h (árvore binária na qual o número de vértices é $2^{h+1} - 1$) está relacionada diretamente com sua altura h.

3.2 Separação de Vértices em Árvores

Concentramos nosso estudo em árvores. Vamos denotar $T[x]$ uma árvore enraizada pelo vértice x e $vs(T[x])$ a separação de vértices dessa árvore. Chamamos de *subárvores induzidas* por um vértice x as subárvores de $T[x]$ que são formadas ao se retirar o vértice x da árvore. As Figuras 3.1 e 3.2 mostram um exemplo de uma árvore e suas subárvores induzidas por um vértice x.

O algoritmo apresentado em [15] enraiza uma árvore T por um de seus vértices e considera recursivamente as subárvores induzidas de nível mais alto (as folhas) até chegar na árvore T de nível mais baixo (nível 0) onde se encontra a raiz, combinando os resultados. Nesse algoritmo são utilizados o conceito de vértice crítico (Definição 3.1), o Teorema 3.1 e o Corolário 3.3, vistos a seguir.

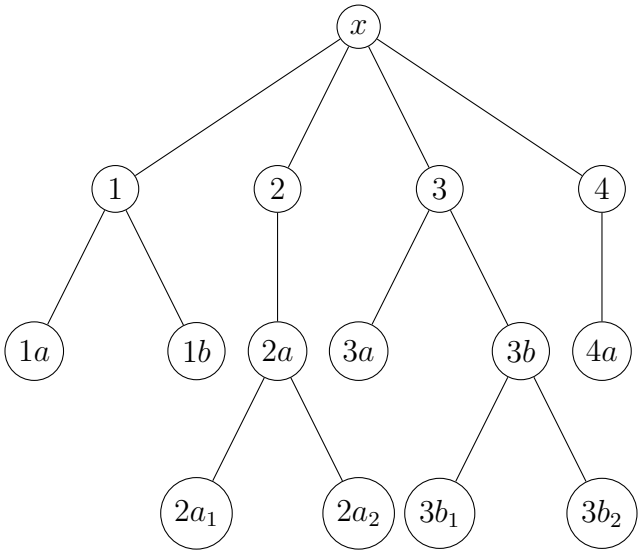

Figura 3.1: Árvore enraizada em x.

Definição 3.1. *Um vértice x é k-crítico em uma árvore enraizada T se e somente se $vs(T[x]) = k$ e existem apenas dois filhos y e z de x tal que $vs(T[y]) = vs(T[z]) = k$.*

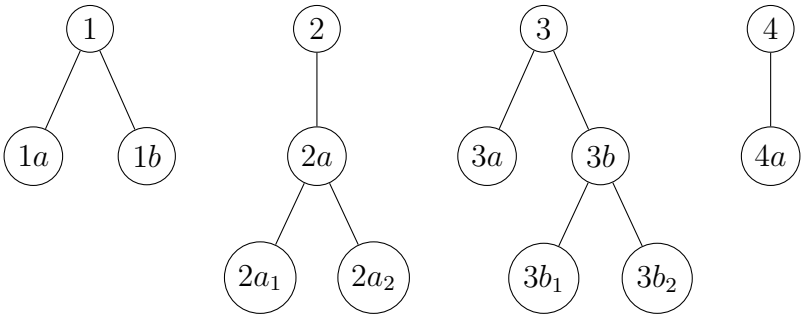

Figura 3.2: Subárvores induzidas pelo vértice x.

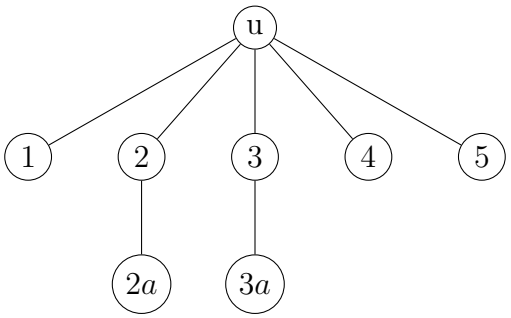

Figura 3.3: Árvore 1-crítico.

O vértice u da Figura 3.3 é um vértice crítico, pois $vs(T[u]) = 1$ e as subárvores $T[2]$ e $T[3]$ induzidas pelo vértice u também possuem separação de vértices 1.

Teorema 3.1. *Seja T uma árvore e seja $k \geq 1$. Então $vs(T) \leq k$ se e somente se para todo vértice x em T no máximo duas subárvores induzidas por x possuiram separação de vértices k e todas as outras subárvores possuem separação de vértices menor ou igual a $k - 1$.*

Vejamos agora um exemplo. Na Figura 3.4, $T[x]$ é uma árvore enraizada pelo vértice x e na Figura 3.5, T_1, T_2 e T_3 são as subárvores induzidas por x. As subárvores T_1 e T_2 possuem separação de vértices igual a 1 e T_3 possui separação de vértices igual a 0. Vemos nesse exemplo que T_1 e T_2 possuem separação de vértices igual a k e T_3 igual a $k - 1$ e $vs(T[x]) = 1$, logo a árvore da Figura 3.4 satisfaz a condição do teorema. Se essa árvore T for enraizada em qualquer um de seus

vértices essa condição continurá a ser satisfeita.

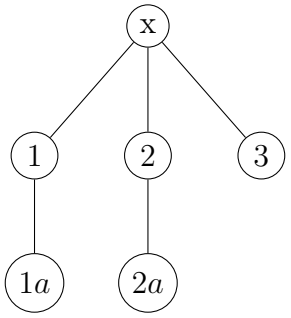

Figura 3.4: Árvore $T[x]$.

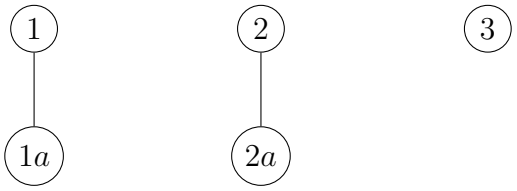

Figura 3.5: Subárvores induzidas por$x$: T_1, T_2 e T_3.

Corolário 3.1. *Temos que $vs(T) > k$ se e somente se existe um vértice que induz três ou mais subárvores T' tal que $vs(T') \geq k$.*

Por exemplo, as subárvores da Figura 3.2 possuem separação de vértices iguais a 1. A árvore $T[x]$, Figura 3.1, possui separação de vértices 2 pois as quatro subárvores induzidas por x possuem a separação de vértices iguais a 1. As quatro subárvores possuem separação de vértices k, logo satisfaz o Corolário 3.1. Por isso $vs(L, G) > k$.

Pelo Teorema 3.1 obtemos que cada árvore T com a separação vértices k tem no máximo um vértice k-crítico. Esse vértice não tem mais do que duas subárvores com separação vértices k, porque, caso contrário, existiria três subárvores de T com separação de vértices k o que contraria o fato de $vs(T) = k$. Assim obtemos:

Corolário 3.2. *Toda árvore T com $vs(T) = k$ possui no máximo um vértice k-crítico.*

Corolário 3.3. *Seja $T[u]$ uma árvore com raiz u dentro de uma árvore enraizada T contendo filhos $v_1, ..., v_d$ e $k = max_i\{vs(T[v_i])\}$. Podemos afirmar que:*

1. *Se mais de duas das árvores $T[v_i]$ possuem separação de vértices k, então $vs(T[u]) = k + 1$.*
2. *Se exatamente duas das árvores $T[v_i]$ possuem separação de vértices k e pelo menos uma contém um vértice k-crítico, então $vs(T[u]) = k + 1$.*
3. *Se exatamente duas das árvores $T[v_i]$ possuem separação de vértices k e nenhuma delas contém um vértice k-crítico, então $vs(T[u]) = k$.*
4. *Se exatamente uma das árvores $T[v_i]$ possui separação de vértices k e ela contém um vértice, x, k-crítico, e $vs(T[u, x]) = k$, então $vs(T[u]) = k + 1$.*
5. *Se exatamente uma das árvores $T[v_i]$ possui separação de vértices k e ela contém um vértice, x, k-crítico, e $vs(T[u, x]) < k$, então $vs(T[u]) = k$.*
6. *Se exatamente uma das árvores $T[v_i]$ possui separação de vértices k e ela não contém um vértice k-crítico, então $vs(T[u]) = k$.*

A inicialização do algoritmo considera que uma árvore com um único vértice tem separação de vértices zero e uma árvore possui separação de vértices igual a 1 se e somente se ela contém no mínimo uma aresta e não contém subárvores com separação de vértices maior que 1.

Como já foi dito anteriormente o algoritmo que acha a separação de vértices para árvores é um algoritmo recursivo. Resumidamente, o algoritmo consiste em enraizar a árvore por um vértice u, andar em suas subárvores até chegar nas folhas, que possuem separação de vértices zero, e de lá voltar localizando a separação de vértices das subárvores $T'[x]$, onde x é a raiz da subárvore, segundo o Corolário 3.3.

A separação de vértices de uma subárvore é determinada da mesma forma que a árvore enraizada. Seja $T[u]$ uma árvore enraizada dentro de T. Seja $T[u, v_1, v_2, ..., v_i]$ a árvore de raíz u da qual foi retirada a subárvore de raíz v_i.

A seguir apresentamos um exemplo:

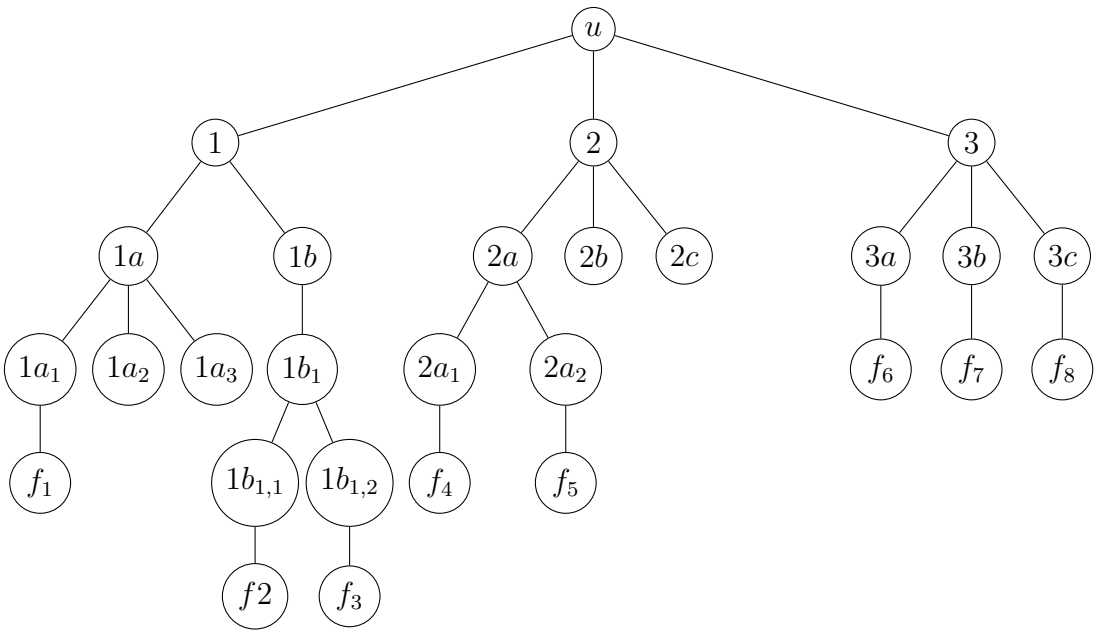

Figura 3.6: Árvore $T[u]$.

Sabemos que a separação de vértices das folhas é sempre zero e dos vértices que só possuem folhas como filhos é um. Portanto:

- $vs(T[f_1]) = vs(T[f_2]) = vs(T[f_3]) = vs(T[f_4]) = vs(T[f_5]) = vs(T[f_6]) = vs(T[f_7]) = vs(T[f_8]) = vs(T[1a_2]) = vs(T[1a_3]) = vs(T[2b]) = vs(T[2c]) = 0.$
- $vs(T[1a_1]) = vs(T[1b_{1,1}]) = vs(T[1b_{1,2}]) = vs(T[2a_1]) = vs(T[2a_2]) = vs(T[3a]) = vs(T[3b]) = vs(T[3c]) = 1.$

Agora vamos determinar a separação de vértices das subárvores de T:

- $vs(T[1a])$:

  A subárvore $T[1a]$ possui três subárvores, porém, apenas uma possue separação de vértices igual a k, que nesse caso $k = 1$. Logo, temos pelo caso 6 $vs(T[a1]) = 1$.

- $vs(T[1b_1])$ e $vs(T[2a])$:

  A separação de vértices de $T[1b_1]$ e $T[2a]$ são determinadas pelo caso 3. Ambas possuem duas subárvores com separação de vértices igual a k, $k = 1$. Logo, temos que $vs(T[1b_1]) = vs(T[2a]) = 1$. Os vértices $1b_1$ e $2a$ são vértices críticos.

- $vs(T[1b])$:

  Após determinar a separação de vértices da subárvore $T[1b_1]$ é a vez de calcular $vs(T[1b])$. Possuindo apenas uma subárvore com separação de vértices k, $k = 1$, e esta contendo um vértice crítico, $1b_1$, também temos que $vs(T[1b, 1b_1]) = 0 < k = 1$. Logo, pelo caso 5, $vs(T[1b]) = 1$.

- $vs(T[1])$:

  Temos que $T[1]$ possui duas subárvores onde $vs(T[1a]) = vs(T[1b]) = 1 = k$. Logo, pelo caso 2, $vs(T[1]) = 2$.

- $vs(T[2])$:

  A subárvore $T[2]$ possui três subárvores, porém, apenas uma possui separação de vértices igual a k, $k = 1$, e essa subárvore possui um vértice crítico. Temos que $vs(T[2, 2a]) = 1 = k$. Logo, pelo caso 4 $vs(T[2]) = 2$.

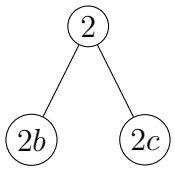

Figura 3.7: $T[2, 2a]$.

- $vs(T[3])$

  Chegamos na última subárvore de $T[u]$. $T[3]$ possui três subárvores com separação de vértices igual a k, $k = 1$. Logo, pelo caso1, $vs(T[3]) = 2$.

Após calcular a separação de vértices de todas subárvores temos que $T[u]$ possui três subárvores. As três possuem separação de vértices 2. Logo, temos pelo caso 1, $vs(T[u]) = 3$.

3.3 Separação de Vértices para Caterpillars

Como um caterpillar é uma árvore, para determinar sua separação de vértices foram utilizados os seis casos descritos no Corolário 3.3 (apresentados em [15]). Foi constatado que independentemente do caterpillar considerado sua separação de vértices é sempre constante, como provado no teorema a seguir.

Teorema 3.2. *Se G é um caterpillar então* $\text{MINVS}(G) = 1$.

Prova. Seja G um caterpillar. Com o intuito de utilizar o Corolário 3.3 é necessário enraizar o caterpillar G por um de seus vértices. Podemos enraizar G pelos vértices da sua espinha dorsal (vértices das extremidades- caso 1 - ou intermediários - caso 2), ou por uma das folhas (caso 3).

Caso 1: Enraizando em v_{n_1} ou em v_1

Suponha o caterpillar G enraizado por v_{n_1}. Para o vértice v_1 temos $vs(T[v_1]) = 1$, pois a separação de vértices das folhas é 0 e uma árvore onde todas suas subárvores induzidas pela raiz são folhas, possui separação de vértices igual a 1.

Considere o vértice v_2 e suas subárvores, $T[v_1]$,$T[w_{2,1}]$,..., $T[w_{2,j}]$. Todas, exceto $T[v_1]$, são folhas. Logo $vs(T[w_{2,1}] =, \ldots, = vs(T[w_{2,j}]) = 0$. Como $T[v_1] = 1$, então $vs(T[v_2]) = 1$ pelo caso 6 do Corolário 3.3, pois ela possui apenas uma subárvore induzida com separação de vértices igual a k.

Por raciocínio análogo, podemos concluir que $T[v_3] = T[v_4] = \ldots = T[v_{n1-1}] = 1$. Por fim, a subárvore $T[v_{n_1-1}]$ é induzida pelo vértice v_{n_1}, onde todas as outras subárvores induzidas por esse vértice são folhas. Logo, $vs(T[v_{n_1}]) = 1$.

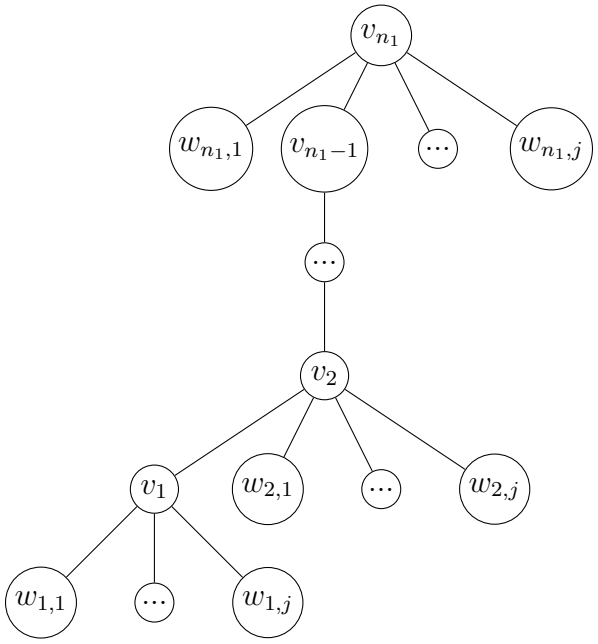

Figura 3.8: Árvore do caterpillar enraizada por v_{n_1}.

Resultado análogo pode ser obtido se o caterpillar G foi enraizado em v_1.

Caso 2: Enraizando em v_i, $v_i \neq v_1$ e $v_i \neq v_{n_1}$

Ao enraizar a árvore por um vértice v_i, $1 < i < n_1$, esse vértice induz duas subárvores que não são folhas, $T[v_{i-1}]$ e $T[v_{i+1}]$. Essas subárvores são idênticas àquela estudada no caso 1. Logo, $vs(T[v_{i-1}]) = vs(T[v_{i+1}]) = 1$. Com isso $T[v_i]$ se

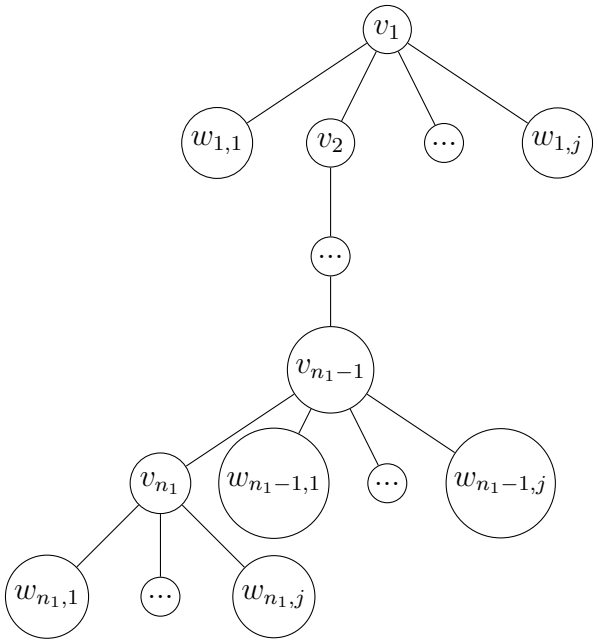

Figura 3.9: Árvore do caterpillar enraizada por v_1.

enquadra no caso 3 do Corolário 3.3, e portanto, $vs(T[v_1]) = 1$.

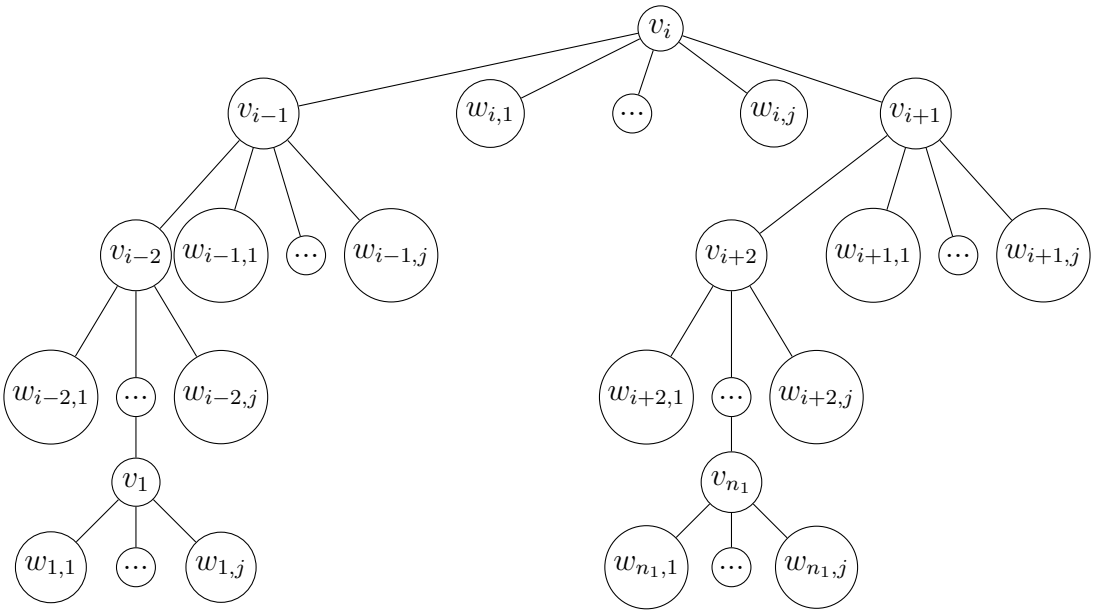

Figura 3.10: Árvore enraizada pelo vértice: v_i, $v_i \neq v_1$ e $v_i \neq v_{n_1}$.

Caso 3: Enraizando por uma folha $w_{i,j}$

Considere a árvore enraizada por uma folha $w_{i,j}$. Como cada $w_{i,j}$ esta ligado a apenas um vértice v_i existe somente uma subárvore induzida pela raiz.

Quando enraizamos por uma folha que esta ligada a um dos vértices das extremidades da espinha dorsal ($i = 1$ ou $i = n_1$) temos um caminho, como no primeiro caso provado.

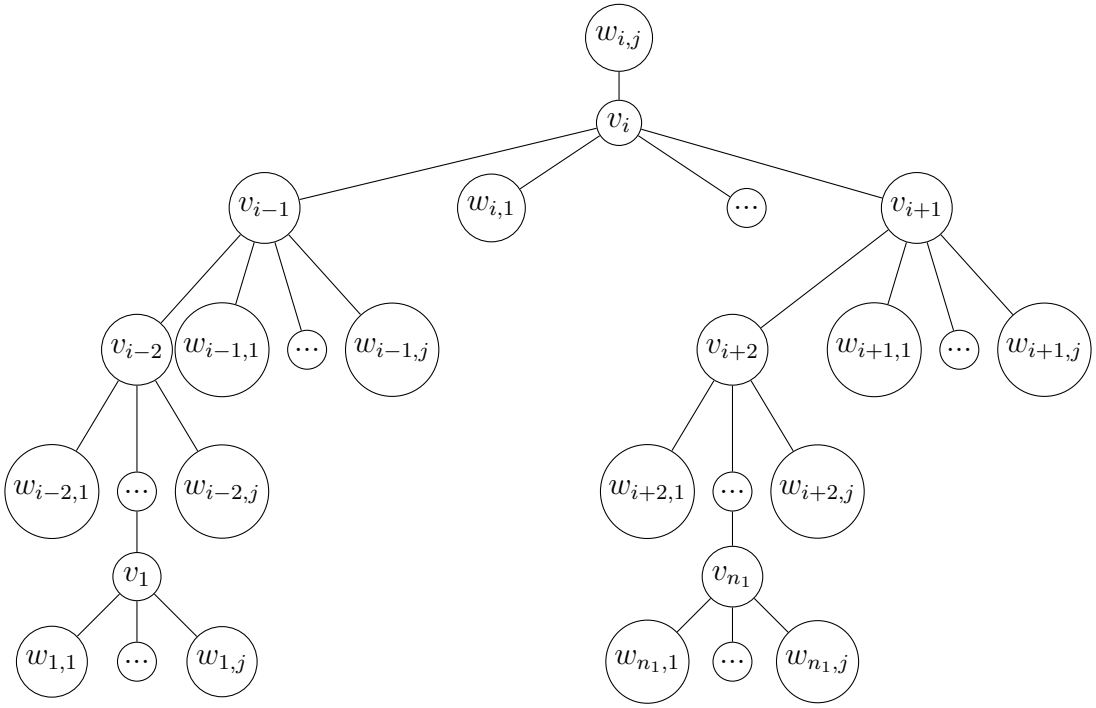

Figura 3.11: Árvore enraizada pelo vértice: folha w_{ij}.

Já ao enraizar por qualquer outra folha, ($i = 2, \ldots, n_1 - 1$), a raíz induz uma subávore, que por sua vez, induz duas subárvores com separação de vértices igual a 1, como a do segundo caso provado. Portanto o vértice ligado à raiz será crítico. Como a raíz possui uma subárvore com separação de vértices igual a 1 e a árvore formada ao se retirar a subárvore enraizada pelo vértice crítico tem separação de vértices menor que 1, então temos o caso 5 do Corolário 3.3. Logo $vs(T[w_{i,j}]) = 1$.

□

É fácil observar o porquê de uma árvore caterpillar nunca cair nos outros casos. Para cair no caso 1 a raiz tem que induzir pelo menos três subárvores com mesma separação de vértices. Isso é impossível de acontecer: devido à definição de caterpillar, a raiz pode induzir muitas subárvores; Mas no máximo duas com a separação de vértices diferente de zero. Isso ocorre pois no caminho, que é onde teremos separação de vértices diferente de zero, só é possível formar no máximo duas subárvores. O caso 2 necessita da existência de um vértice crítico em pelo menos uma das duas subárvores induzidas pela raiz. Como vimos isso é impossível devido à estrutura do caterpillar. Já o caso 4 nunca ocorrerá pois, ao retirar o vértice crítico, é impossível a separação de vértices de $T[w_{i,j}, v_i]$ ser igual a $vs(T[w_{i,j}])$.

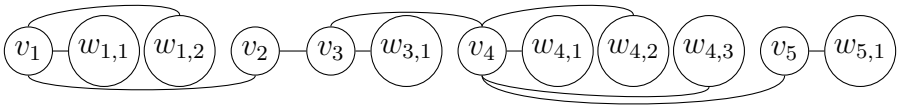

Figura 3.12: Layout ótimo do caterpillar.

A Figura 3.12 mostra um dos layouts ótimos do caterpillar da Figura 1.1. Observe que o layout tem na primeira posição o primeiro vértice da espinha dorsal, v_1. Nas posições seguintes estão todas as folhas de v_1. Em seguida vem v_2, v_3 e todas as folhas de v_3. Essa ordenação de vértice da espinha dorsal seguido de suas folhas segue até o fim do layout. Temos então uma regra para determinar um dos layouts ótimos de um caterpillar: um vértice da espinha dorsal seguido de suas folhas adjacentes, começando o layout por um dos vértices das extremidades e respeitando a ordem de adjacência da espinha dorsal. É imediato concluir que esta construção pode ser efetuada com complexidade de tempo de $O(n)$.

4 ARRANJO LINEAR MÍNIMO

4.1 Introdução

O problema do arranjo linear mínimo foi primeiramente apresentado por Harper [21], com o objetivo de estudar códigos de correção de erros. O problema do arranjo linear mínimo tem recebido alguns nomes alternativos: *melhor ordenação linear, soma das arestas, 1-soma mínima, soma da largura de banda* ou *problema comprimento do fio.*

O arranjo linear de um layout LA é a soma do comprimento de todas suas arestas. O problema de determinar o arranjo linear mínimo, consiste em dado um grafo G, encontrar o layout ótimo, ou seja, o layout que possui o menor arranjo linear entre todos os layouts de G.

Seja $LA(L,G) = \sum_{uv \in E} \lambda(uv, L, G)$. Dado $G = (V, E)$, o problema da *determinação do arranjo linear mínimo* consiste em encontrar um layout L^* tal que

$$LA(L^*, G) = \min_{L \in \mathcal{L}} LA(L, G),$$

sendo $\mathcal{L}$ o conjunto de todos os layouts de G. O valor $LA(L^*, G)$ é denotado $\text{MINLA}(G)$.

Existe uma conexão entre o problema do número de cruzamento bipartido *(bipartite crossing number problem)* e o problema do arranjo linear mínimo. O problema do número de cruzamento bipartido é a determinação do menor número

de cruzamento de arestas entre todos os traçados bipartidos de G. Shahrokhi [48] mostrou um algoritmo em que o arranjo linear mínimo pode se usado para localizar o número de bipartição de uma árvore, demonstrando assim que um algoritmo com complexidade de $O(n^{1.6})$ existe para esse problema. Em [23], é apresentado um algoritmo em tempo linear para uma classe restrita de grafos chamada *one-page*. Um layout desse tipo acontece se não existirem quatro vértices a, b, u, v onde (a, b) e (u, v) são arestas e $L(a) < L(u) < L(b) < L(v)$, ou seja, sem que existam arestas cruzadas no layout. Para grafos de intervalo, [11] prova que o arranjo linear mínimo é NP-difícil. O mesmo resultado é válido para grafos de permutação. É apresentado um limite inferior e um algoritmo simples e rápido, com aproximação de um fator 2, com base em qualquer modelo de grafo intervalo. Um algoritmo polinomial para a atribuição de um grafo série-paralelo é sugerido por [58].

4.2 Arranjo Linear Mínimo para Árvores

Para árvores, Diáz fala em [12] que Goldberg and Klipker [19] mostraram um algoritmo com complexidade de tempo $O(n^3)$ para resolver o problema do arranjo linear mínimo. Em 1979, Shiloach [49] provou um algoritmo complexidade $O(n^{2.2})$ para árvores não direcionadas. Em seu artigo Shiloach define três tipos de layouts:

Seja v_* um vértice que induz as subárvores $T_1, T_2, ..., T_k$,.

- **Tipo 1:** $(T_1, T_2, ..., T_k|v_*)$

  O layout L é do tipo $(T_1, T_2, ..., T_k|v_*)$ se a seguinte relação se mantém:

$$\sum_{i=1}^{j-1} n_i < L(v) \leq \sum_{i=1}^{j} n_i < L(v_*),$$

  para todo $v \in T_j$ e para todo $1 \leq j \leq k$, onde n_i denota o número de vértices

de T_i, $i = 1, ..., k$.

- **Tipo 2:** $(v_*|T_1, T_2, ..., T_k)$

  O layout L é do tipo $(v_*|T_1, T_2, ..., T_k)$, se o inverso do layout L, $\overline{L}$, for do tipo $(T_1, T_2, ..., T_k|v_*)$.

- **Tipo 3:** $(T_1, T_2, ..., T_p|v_*|T_{p+1}, ..., T_k)$

  O layout L é do tipo $(T_1, T_2, ..., T_p|v_*|T_{p+1}, ..., T_k)$ se for dos tipos 1 e 2, $(T_1, T_2, ..., T_k|v_*)$ e $(v_*|T_1, T_2, ..., T_k)$, ao mesmo tempo.

Os três tipos são ilustrados na Figura 4.1.

Basicamente, esses tipos de layouts são três formas de organizar as subárvores de um determinado vértice v_*.

Também são definidos dois conceitos de árvores: árvore ancorada e livre. Árvores ancoradas são subárvores $(T_1, ..., T_k)$ que pertencem à árvore T, induzidas por um v_*, onde, v_* liga essas subárvores a árvore T. A diferença entre uma árvore ancorada e uma árvore livre, é que na primeira, v_* é o vértice que liga a âncora a árvore T, já na segunda, v_* é à raiz da árvore T.

O algoritmo é recursivo e trata as árvores ancoradas e livres simultaneamente, usando um parâmetro α. Para árvores livres, $\alpha = 0$, e para árvores ancoradas, $\alpha = 1$. Mesmo quando a árvore é livre ela é ancorada pelo algoritmo, ou seja, a árvore é enraizada pelo v_* e seus filhos são divididos em duas partes, a parte à direita e à esqueda de v_*.

Primeiramente, o vértice v_* é determinado através do teorema do vértice central.

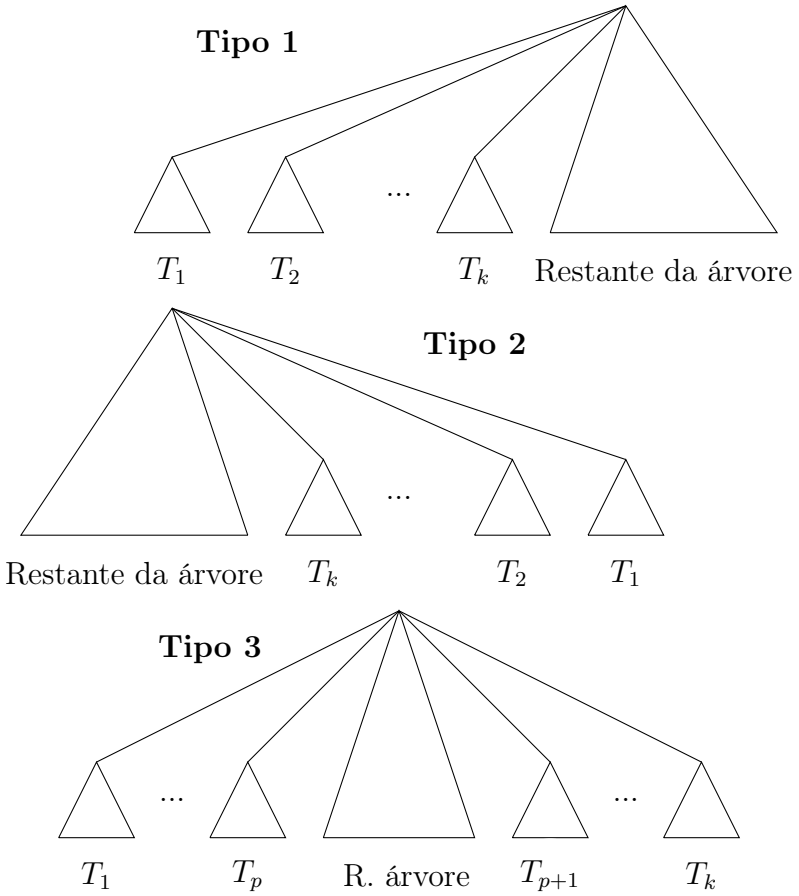

Figura 4.1: Tipos de layout.

Teorema 4.1. *Teorema do Vértice Central: Existe um vértice v_* satisfazendo: se $T_0, ..., T_k$ são todas as subárvores induzidas por v_* então:*

$$n_i \leq \left\lfloor \frac{n}{2} \right\rfloor, i = 1, ..., k.$$

As subarvores $T_0, ..., T_k$, induzidas por v_*, são numeradas tal que $n_0 \geq n_1 \geq ... \geq n_k$. Em seguida é determinado o layout ótimo para as âncoras direita e esquerda. As âncora direita e esquerda são definadas pelos seguintes custos:

$$\textit{âncora direita: } C[L, T_D(v)] = C[L, T] + n - L(v);$$

$$\textit{âncora esquerda: } C[L, T_E(v)] = C[L, T] + L(v) - 1.$$

Em seguida é calculado o custo, $C_\alpha(A)$, para um layout do tipo $(T_0|v_*)$. Para árvores livres, $C_\alpha(A)$ é a soma do custo da parte direita com o custo da parte esquerda da árvore, mais 1. Para árvores ancoradas, é a soma do custo da âncora direita, mais o custo do restante da árvore, mais o número de vértices da árvore, menos o número de vértices da âncora direita.

O próximo passo é calcular p_α, que é o maior inteiro que satisfaz a seguinte inequação:

$$n_i > \left\lfloor \frac{n_* + 2}{2} \right\rfloor + \left\lfloor \frac{n_0 + 2}{2} \right\rfloor,$$

para $i = 1, ..., 2p_\alpha - \alpha$, onde

$$n_* = n - \sum n_i.$$

Se $p_\alpha = 0$ então o $C_\alpha(A)$ é a resposta do problema. Caso contrário, $p_\alpha > 0$, o arranjo linear mínimo pode ser de dois tipos: $(T_0|v_*)$ ou $(T_1, T_3, ..., T_{2p_\alpha - 1}|v_*|T_{2p_\alpha - 2\alpha}, ..., T_4, T_2)$. Então, é calculado o custo para esse segundo tipo, $C_\alpha(B)$. Se $C_\alpha(A) < C_\alpha(B)$ então $C_\alpha(A)$ é o arranjo linear mínimo. Se não o arranjo linear mínimo é $C_\alpha(B)$.

Os custos $C_\alpha(A)$ e $C_\alpha(B)$, são calculados pelos teoremas 4.2 e 4.3. O primeiro teorema diz qual o tipo do layout. O segundo mostra como calcular o custo para cada tipo de layout com seu respectivo α.

Teorema 4.2. *a) Se $p_\alpha = 0$ então $T(\alpha)$ possui o layout mínimo do tipo $(v_0|v_*)$. b) Se $p_\alpha > 0$ então $T(\alpha)$ possui o layout mínimo de um dos tipos asseguir:*

$$A\text{: } (T(v_0|v_*))$$

$$B\text{: } (T_1, T_3, ..., T_{2p_\alpha-1}|v_*|T_{2p_\alpha-2\alpha}, ..., T_4, T_2)$$

Teorema 4.3. *a) Se L é um layout de $T(\alpha)$ do tipo A então*

$$C_\alpha(A) = C[L, T(\alpha)] = \begin{cases} C[L, (T_0)_D(v_0)] + C[L, (T - T_0)_E(v_*)] + 1 & se \quad \alpha = 0. \\ C[L, (T_0)_D(v_0)] + C[L, T - T_0] + n - n_0 & se \quad \alpha = 1. \end{cases}$$

b) Se L é um layout de $T(\alpha)$ do tipo B então

$$C_\alpha(B) = C[L, T(\alpha)] = \sum_{i=1}^{2p_\alpha-1} C[L, (T_i)_D(v_i)] + \sum_{i=1}^{2p_\alpha-2\alpha} C[L_i, (T_i)_E(v_*)] + C[L, T_*] + S_\alpha.$$

Seja $T_* = T(\alpha) - (T_1, ..., T_{2p_\alpha-2\alpha})$ e seja:

$$S_0 = (n_3 + n_4) + 2(n_5 + n_6) + ... + (p_0 - 1)(n_{2p_0-1} + n_{2p_0}) + p_0(n_* + 1),$$

$$S_0 = (n_2 + n_3) + 2(n_4 + n_5) + ... + (p_1 - 1)(n_{2p_1-1} + n_{2p_1}) + p_1(n_* + 1) - 1.$$

Ainda para árvores, em 1983, Chung em [10], prova um algoritmo com complexidade $O(n^\lambda)$ para λ satisfazendo $\lambda > \frac{\log 3}{\log 2}$, aproximadamente 1.6. O artigo dado por Chung, primeiramente, analiza um algoritmo $O(n^2)$, que é similar ao algoritmo apresentado em [49]. Em seguida é dado o algoritmo $O(n^{\frac{log3}{log2}})$. O segundo algoritmo é uma versão refinada do primeiro mostrado. A principal ideia é encontrar o arranjo linear mínimo de forma mais eficiente nesse processo recursivo. O algoritmo consiste em três partes. A primeira determina o arranjo linear mínimo para uma árvore T. A segunda parte determina o arranjo linear mínimo para uma árvore enraizada T^*.

E a terceira parte, para duas árvores, T^* e $\overline{T}^*$ com $|V(T)| \geq |V(\overline{T})|$, determina o custo do arranjo linear mínimo para T^* e $T \cup \overline{T^*}$, que é a árvore formada pela combinação de T e $\overline{T}$ com uma aresta ligando as duas raizes de T e $\overline{T^*}$.

4.3 Arranjo Linear Mínimo para Caterpillar

Apresentaremos uma forma mais simples de encontar o arranjo linear mínimo para a classe dos caterpillars, Mostrando que no layout ótimo as arestas devem possuir o menor comprimento possível.

Primeiramente vamos tratar dos dois casos particulares: os grafos caminho e estrela.

Caso 1: Grafo caminho

Teorema 4.4. *Se G é um caminho, então* $\text{MINLA}(G) = |E| = |V| - 1$.

Prova. Seja G um grafo caminho. Logo, G é conexo. Para n vértices, o menor número de ligações possíveis é $n - 1$ arestas. A distância entre dois vértices adjacentes é no mínimo 1. Se todos os vértices adjacentes possuirem distância 1 então o somatório das distâncias entre dois vértices adjacentes será no mínimo $n - 1$. Logo, $\text{MINLA}(G) = n - 1$. □

A Figura 4.2 ilustra o resultado do Teorema 4.4.

Caso 2: Grafo estrela

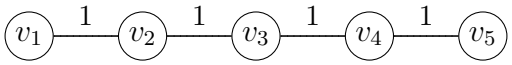

Figura 4.2: Exemplo de um grafo caminho.

Um grafo estrela é uma árvore onde um único vértice tem grau diferente de 1 e está ligado a todos os outros vértices. O vértice central, v_c, do grafo estrela é o único vértice que possui grau diferente de um. Para um layout ótimo, v_c deve ficar o mais central possível, pois assim as arestas terão o menor comprimento e o layout será ótimo. Logo, a posição do vértice central no layout é dada por:

$$L(v_c) = \begin{cases} \lfloor \frac{n}{2} \rfloor + 1, & \text{se } n \text{ for ímpar,} \\ \frac{n}{2}, & \text{se } n \text{ for par.} \end{cases}$$

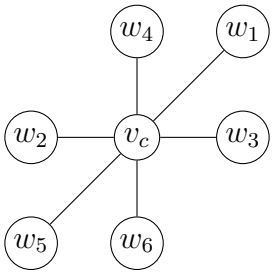

Figura 4.3: Grafo estrela.

O arranjo linear mínimo do grafo estrela é o somatório das distâncias do vértice central até todos os vértices à sua esquerda, mais o somatório das distâncias do vértice central até todos os vértices à sua direita. Caso o número de vértices do grafo estrela seja ímpar existirá a mesma quantidade de vértices à direita, n_d, e à esquerda, n_e, do vértice central. Caso contrário um dos lados ficará com um vértice a mais. O arranjo linear mínimo é dado por:

$$\sum_{uv \in E} \lambda(uv, L, G) = \sum_{uv \in \mathcal{D}(i,L,G)} \lambda_d(uv, L, G) + \sum_{uv \in \mathcal{E}(i,L,G)} \lambda_e(uv, L, G)$$

onde, $\lambda_d(uv, L, G)$ e $\lambda_e(uv, L, G)$ são as distâncias do vértice central até um vértice da direita e da esquerda respectivamente.

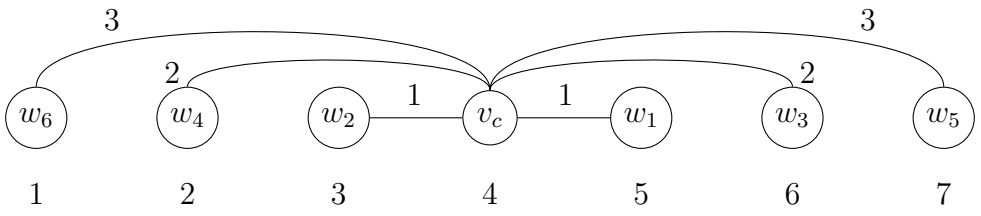

Figura 4.4: Layout do grafo estrela com n ímpar.

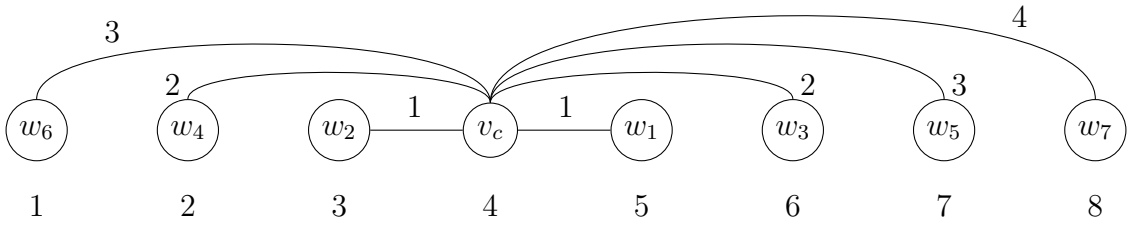

Figura 4.5: Layout do grafo estrela com n par.

Quando n for ímpar, $\lambda_d(uv, L, G) = \lambda_e(uv, L, G)$. Nos casos em que n for par, a distância do vértice mais afastado do centro é dada por: $|\lambda_d(uv, L, G) - \lambda_e(uv, L, G)|$. O arranjo linear mínimo da metade do layout é a soma de uma progressão aritmética de razão 1, $S_p = \frac{p(a_1+a_p)}{2}$ onde, a_1 é 1 (distância do vértice mais próximo do centro), a_p é a distância do mais distante e p é o número de vértices do lado do layout que está sendo analizado. Logo, o arranjo linear mínimo para estrelas com n ímpar é dado por:

$$\sum_{uv \in E} \lambda(uv, L, G) = \frac{2\left(\frac{(n-1)}{2}\right)\left(1 + \frac{(n-1)}{2}\right)}{2} = \frac{n^2 - 1}{4},$$

e para n par é dado por:

$$\sum_{uv \in E} \lambda(uv, L, G) = \frac{\frac{n}{2}\left(1 + \frac{n}{2}\right)}{2} + \frac{\left(\frac{n}{2} - 1\right)\left(1 + \left(\frac{n}{2} - 1\right)\right)}{2} = \frac{n^2}{4}.$$

Corolário 4.1. *Seja G um grafo estrela. Então o arranjo linear mínimo de um*

grafo estrela é dado por:

$$\text{MIMLA}(G) = \begin{cases} \dfrac{n^2-1}{4}, & \textit{para} \quad n \quad \textit{ímpar}, \\ \dfrac{n^2}{4}, & \textit{para} \quad n \quad \textit{par}. \end{cases}$$

Temos então, para o grafo da Figura 4.4, MIMLA(G) = 12 e para o grafo da Figura 4.5, MIMLA(G) = 16.

Caso 3: Caterpillar

O arranjo linear mínimo para caterpillar foi dado por Horton em [24]. Ele prova o arranjo linear mínimo para caterpillars como parte da resolução do problema de determinar o arranjo linear mínimo para a classe dos grafos Halin. *Grafos Halin* são grafos planares com $n > 3$, com a propriedade de que o conjunto de arestas pode ser dividido em uma árvore, onde nehum vértice possui grau menor do que 3, e um ciclo C, que une todas as folhas da árvore. Para provar o arranjo linear mínimo para caterpillars, Horton utiliza o Teorema 4.5, que determina o limite inferior do arranjo linear mínimo para um caterpillar dado, e a Definição 4.2.

Propriedade 4.1. *Os vértices $w_{1,1}$ e $w_{n_1,j}$ com $L(w_{1,1})$ e $L(w_{n_1,j})$ rotulados com 1 e n respectivamente, são folhas, isto é, $d(w_{1,1}) = d(w_{n_1,j}) = 1$.*

Propriedade 4.2. *Seja P um caminho no caterpillar G formado pela espinha dorsal acrescentados dos vértices $w_{1,1}$ e $w_{n_1,j}$ rotuladas por 1 e n. O caminho P é denotado por $i_0, ..., i_{n_1+1}$, onde i_j é o $(j+1)$-ésimo vértice do caminho. Então os rótulos de P são "monotonos" tais que:*

$$L(i_j) < L(i_{j+1}) \text{ para } i = 0, 1, ..., n_1,$$
$$L(i_j) > L(i_{j+1}) \text{ para } i = 0, 1, ..., n_1.$$

Propriedade 4.3. *Suponha P um caminho conectando um par de folhas no caterpillar e, além disso, seja esse o maior caminho em T. Então o grafo formado pela remoção das arestas do caminho P é uma coleção de estrelas disjuntas onde cada uma está rotulada por números inteiros consecutivos.*

Teorema 4.5. *Seja G um caterpillar com n vértices, sendo v_i, $i = 1, 2, ..., n_1$ os vértices da espinha dorsal do caterpillar. Então*

$$\text{MINLA}(G) \geq n - 1 + \sum_{i=1}^{n_1} \left\lfloor \frac{(d(v_i) - 1)^2}{4} \right\rfloor.$$

Para calcular o arranjo linear mínimo de um caterpillar G, primeiramente, retiramos as arestas que formam a sua espinha dorsal. Ao retirar as arestas, G é separado em várias estrelas.

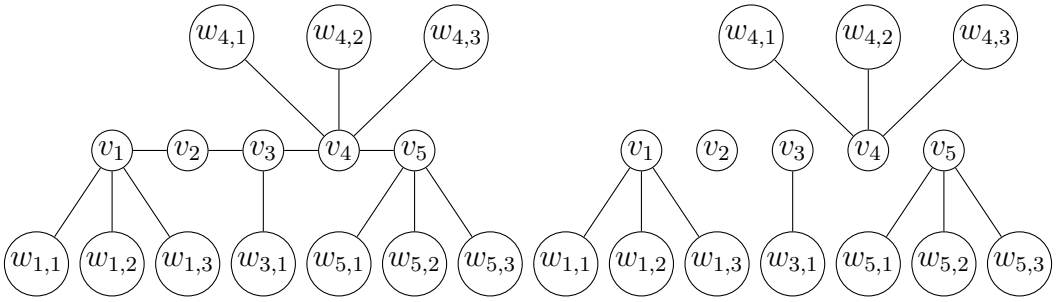

Figura 4.6: Conjunto das estrelas contidas no caterpillar G.

Em seguida calculamos o arranjo linear mínimo e o layout ótimo para cada estrela contida no caterpillar. Como queremos ter as arestas com o menor comprimento possivel, ao separar o caterpillar em grafos estrelas e achar o melhor layout

desse grafo, temos a menor distância entre um vértice da espinha dorsal e suas folhas. O segundo passo é arrumar o layout dessas estrelas no layout do caterpillar, de forma que os vértices adjacentes na espinha dorsal fiquem o mais próximo possível.

Entre os vértices da espinha dorsal a menor distância possível entre eles é 1. Porém no layout ótimo dois vértices da espinha dorsal só estarão lado a lado se não possuirem folhas, ou no máximo uma folha cada um.

No caso de existir uma quantidade diferente de vértices entre os dois lados do layout da estrela a posição deles no layout do cartepillar só interfere no resultado nas extremidades. Nesse caso, o maior número de folhas deve ficar sempre para fora do layout do caterpillar. Ou seja, se a primeira estrela do layout possuir um número par de vértices, a maior quantidade de folhas ficará no lado esquerdo do layout do caterpillar; se a última estrela do layout do caterpillar possuir número par de vértices, a maior quantidade de folhas ficará do lado direito do layout do caterpillar.

O caterpillar da Figura 4.6 possui $\text{MINLA}(G) = 24$. As Figuras 4.7, $LA(L, G) = 24$, e 4.8, $LA(L, G) = 25$, mostram um layout ótimo e um não ótimo, respectivamente, para o caterpillar da Figura 4.6. O layout não ótimo possui o maior número de folhas, adjacentes aos vértices das extremidades da espinha dorsal, para dentro do layout.

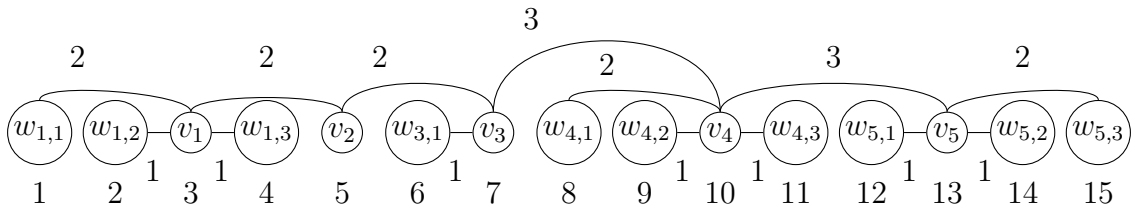

Figura 4.7: Layout ótimo.

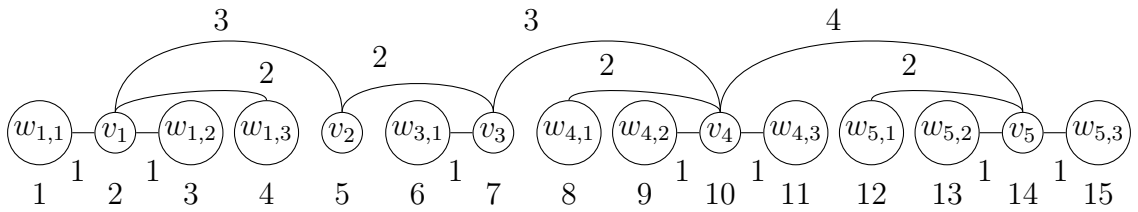

Figura 4.8: Layout não ótimo.

Horton fala que os rótulos dos vértices do caminho do layout satisfazem a "monotocidade" atribuida pela Definição 4.2 e, além disso, o arranjo linear mínimo possui valor de exatamente $n-1$. Cada uma das estrelas, formadas pela remoção das arestas que formam a espinha dorsal, são rotuladas por números inteiros consecutivos e com o rótulo ótimo em cada caso. Tem-se então que o valor da rotulagem total é exatamente o valor limite do Teorema 4.5 e é, portanto, ótimo.

Neste capítulo tratamos do problema da determinação do valor do arranjo linear mínimo para um caterpillar. Também definimos o formato do layout ótimo. No próximo capítulo vamos tratar da largura de banda.

5 LARGURA DE BANDA

5.1 Introdução

O problema da largura de banda surgiu a partir da computação com matrizes simétricas e esparsas, onde as operações são executadas de maneira mais eficiente com a permutação das linhas e colunas da matriz. Define-se largura de banda de uma matriz simétrica esparsa M como o maior natural k para o qual existe um elemento não nulo em $M_{i,i+k}, i \in N$.

Seja $BW(L,G) = \max_{uv \in E} \lambda(uv, L, G)$. Dado um grafo $G = (V, E)$ o problema da *determinação da largura de banda* consiste em encontrar um layout L^* tal que

$$BW(L^*, G) = \min_{L \in \mathcal{L}} BW(L, G),$$

sendo $\mathcal{L}$ o conjunto de todos os layouts de G. O valor $BW(L^*, G)$ é denotado $\text{MINBW}(G)$.

A largura de banda é um problema de referência, frequentemente estudado. Esse problema, em geral, é NP-completo [41]. Porém, existem algoritmos polinomiais para resolver o problema de largura de banda quando restrito a classes particulares de grafos, como: *os caterpillars com cabelos de comprimento de no máximo 2* [3], *grafos intervalo* [52] e *grafos de cadeia* [31]. Existe um levantamento de alguns algoritmos polinomias para essas classes de grafos em [47].

Para os grafos de intervalo, existem dois algoritmos com complexidade de

tempo polinomal: um $O(nk)$ [29] e outro $O(n \log n)$ [52], onde $k \in \mathbb{N}$. O segundo algoritmo é baseado no primeiro. Nessa nova versão, foram feitas mudanças apenas nas estruturas de dados usadas; o algoritmo em si permaneceu o mesmo. Esta nova abordagem traz um limite superior menor para o problema quando se considera valores grandes de k comparados a $\log n$. O primeiro algoritmo, que possui complexidade de tempo polinomial, que calcula a largura de banda de grafos bipartidos e de permutação é dado por Heggernes, Kratsch e Meister [22]. A complexidade de tempo desse algoritmo é de $O(n^4 \log n)$. O algoritmo baseia-se nas propriedades estruturais.

Também existe um algoritmo com complexidade de tempo polinomial para os cografos [56]. Esse algoritmo calcula a largura de banda de G ao ser aplicado a sua coárvore. Já para os grafos de cadeia, um algoritmo com complexidade de tempo $O(n^2 \log n)$ é apresentado por Kloks *et al.* [31].

Para árvores esse problema é NP-completo, porém existem estudos para a classe dos caterpillars, que apresentam algoritmos polinomiais para carterpillars com comprimento de cabelo de no máximo 2.

5.2 Largura de Banda para Caterpillars

Seja G um caterpillar. Ao se substituir uma folha de G por um caminho com c vértices obtem-se uma generalização de caterpillar chamado *caterpillar com comprimento de cabelo maior que* 1.

A largura de banda em grafos caterpillar com cabelos de tamanho no máximo 2 possui um algoritmo com complexidade de tempo polinomial $O(n \log n)$. Em 1981, Assmann, Peck, Syslo e Zak [3] apresentaram um algoritmo de rotulação para um

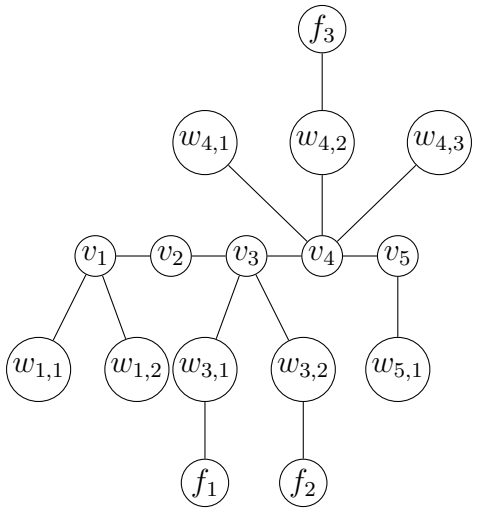

Figura 5.1: Caterpillar com comprimento de cabelo igual a 2.

caterpillar G, com comprimento de cabelo no máximo 2, em que um valor k de largura de banda é testado. Se a largura de banda é menor ou igual a k, então o algoritmo consegue uma rotulação f do grafo G tal que a largura de banda dessa rotulação é igual a k. Se a largura de banda for maior que k, é demonstrado no artigo que o algoritmo é interrompido antes de rotular todos os vértices do grafo e que, nesse caso, existe um subgrafo H de G tal que a largura de banda de H é maior que k. Em 1986, Monien [36] apresentou uma demonstração da NP-completude do problema de largura de banda 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.

Em 1982, Syslo e Zack publicaram outro artigo [53], onde além de definirem a largura de banda para a classe dos caterpillars, eles também nos dão um algoritmo polinomial com complexidade de tempo $O(n^2)$, que monta o layout ótimo para o caterpillar dado. Os resultados deste artigo, apresentados a seguir, servirão também de base para Seção 5.3.

Seja T um caterpillar. Seja $T_{ip}(i \leq p)$ um subgrafo de T formado por $\{v_i, v_{i+1}, ..., v_p\}$, $1 \leq i \leq p \leq n_1$ e todos os seus vizinhos. A largura de banda de T é definida pelo seguinte teorema:

Teorema 5.1. *Se T é um caterpillar com n_1 vértices na espinha dorsal, então*

$$BW(L^*, T) = \max_{1 \leq i,p \leq n_1} b(T_{ip})$$

onde, $b(T_{ip}) = \left\lceil \frac{n(T_{ip}) - 1}{diam(T_{ip})} \right\rceil$ *e* $n(T_{ip})$ *é o número de vértices de* T_{ip}.

A Figura 5.2 mostra um caterpillar e um de seus subgrafos T_{ip}.

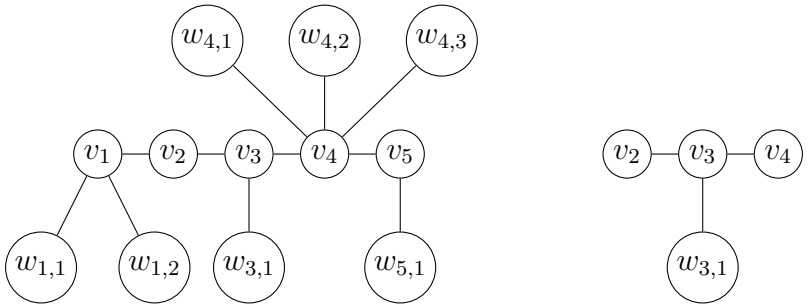

Figura 5.2: Caterpillar e o subgrafo T_{33}.

A seguir mostramos o algoritmo dado por Syslo e Zack em [53] para o cálculo da largura de banda e a formação do layout ótimo. No primeiro passo é determinada a largura de banda considerando o Teorema 5.1. No próximo passo é encontrado o layout ótimo.

O algoritmo determina a rotulação de cada uma das estrelas contidas no caterpillar. Após a rotulação de uma estrela, é armazenado na variável $maior_L$ o maior rótulo utilizado até o momento por um vértice. Para determinar $L(v_i)$ o algoritmo executa dois cálculos, e o menor resultado entre os dois é utilizado. Para $L(w_{ij})$ são utilizados os inteiros que ainda não foram utilizados do conjunto $[1, n]$.

Seja $F(v_i)$ o conjunto de todas as folhas adjacentes a v_i. O passo inicial é pegar uma folha de v_1 e a rotular 1, isto é, $L(w_{1,1}) = 1$. Feito isso o algoritmo desconsidera essa folha no conjunto de folhas de v_1, $F(v_1)$.

Algoritmo $Largura\_de\_Banda$;
Entrada: Caterpillar G.
Saída: Layout ótimo do caterpillar G.
início
% Passo inicial:
Seja w_{11} uma folha ligada a v_1, Seja $v_0 = w_{11}$. $L(v_0) \leftarrow 1$;
$maior_L \leftarrow 1$, $BW \leftarrow 0$;
$num\_elem(F(v_1)) \leftarrow num\_elem(F(v_1)) - 1$;
% Determinando a largura de banda:
para $i \leftarrow 1, \ldots, n_1$ **faça**
para $j \leftarrow i, \ldots, n_1$ **faça**
$b \leftarrow \left\lceil \frac{n(T_ip)-1}{diam(T_{ip})} \right\rceil$;
se $b > BW$ **então**
$BW \leftarrow b$;
% Determinando o layout ótimo:
para $i \leftarrow 1, \ldots, n_1$ **faça**
$L_1 = L(v_{i-1}) + BW$;
$L_2 = maior_L + num\_elem(F(v_i)) + 1$;
se $L_1 < L_2$ **então**
$L(v_i) \leftarrow L_1$;
se não
$L(v_i) \leftarrow L_2$;
Numerar todas $w_{ij} \in v_i$ com os menores inteiros não usados do intervalo $[1, n]$;
$maior_L$ recebe o maior $L(w_{ij})$;
fim.

A Figura 5.3 foi obtida através do algoritmo; ela é o layout ótimo para largura de banda do cartepillar da Figura 5.2.

Podemos observar que quando tivermos $p = i$ o subgrafo T_{ip} será uma estrela,

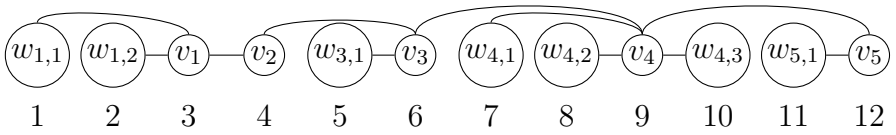

Figura 5.3: Layout ótimo do caterpillar para largura de banda.

T_{ii}. A partir do Teorema 5.1 nós provamos o Corolário 5.1 que determina a largura de banda de uma estrela. Este corolário será muito importante na próxima seção.

Corolário 5.1. *Seja G uma estrela com n vértices.*

$$\text{MINBW}(G) = \left\lceil \frac{n-1}{2} \right\rceil.$$

Prova. Sabemos como calcular a largura de banda para um caterpillar. Um grafo estrela possui apenas T_{11}. Logo, $n(T_{11}) = n$ e o diâmetro de uma estrela é sempre 2. □

5.3 Caterpillar 2-estrelas

Alguns caterpillars possuem propriedades específicas. Seja T um caterpillar, com os vértices da espinha dorsal denotados de v_1 até v_{n_1}, onde o primeiro e o último vértice da espinha dorsal, v_1 e v_{n_1}, possuem k e l folhas adjacentes respectivamente, sendo $k + l = n_2$ e $k + l > 1$, onde $k, l \geq 2$. Chamamos esse caso particular de caterpillar de *caterpillar 2-estrelas*, denotando $C_{k,l}$.

É interessante observar que esta subclasse de caterpillar foi definida por Fallat e Kirkland [17], no contexto da determinação das conectividades algébricas extremas

em árvores, da seguinte forma: seja P_{d_1} um caminho com comprimento $d-2$ rotulado de 1 até $d-1$. Seja $T_d(k,l)$ uma árvore com n vértices e diâmetro d obitida de P_{d-1} adicionando-se k vértices pendentes ao vértice 1 e l vértices pendentes ao vértice $d-1$. Sendo $k, l \geq 2$.

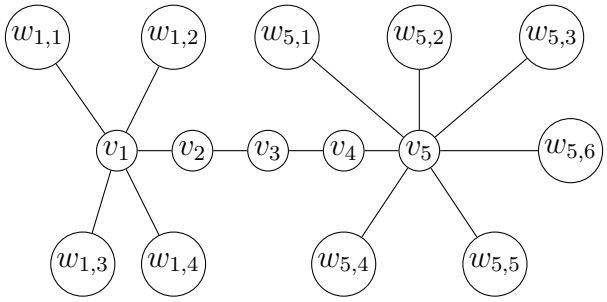

Figura 5.4: Caterpillar 2-estrelas.

Teorema 5.2. *Seja $G = C_{k,l}$ com $n_1 \geq 3$. A largura de banda de G é:*

$$\text{MINBW}(G) = \max\left(\left\lceil \frac{n(T_{11})-1}{diam(T_{11})} \right\rceil, \left\lceil \frac{n(T_{n_1 n_1})-1}{diam(T_{n_1 n_1})} \right\rceil\right).$$

Prova. Seja G um caterpillar 2-estrelas $C_{k,l}$, $n_1 \geq 3$.

No layout ótimo do caterpillar $C_{k,l}$, os vértices da espinha dorsal entre v_1 e v_{n_1} não interferem no maior comprimento de aresta. Isso porque como esses vértices não possuem folhas adjacentes a eles, as arestas (v_i, v_{i+1}), para $i = \{2, ..., n_1 - 2\}$ sempre irão possuir comprimento igual a 1.

Vamos começar calculando $b(T_{ip})$ para T_{11}. O número de vértices de T_{11} é igual ao número de folhas, k, mais dois, que são v_1 e v_2,

$$b(T_{11}) = \left\lceil \frac{n(T_{11})-1}{2} \right\rceil = \left\lceil \frac{k+2-1}{2} \right\rceil = \left\lceil \frac{k+1}{2} \right\rceil.$$

Primeiramente vamos provar que $b(T_{11}) \geq b(T_{1p})$, para p variando de 2 até $n_1 - 1$.

$$b(T_{1p}) = \left\lceil \frac{k + (p+1) - 1}{p+1} \right\rceil = \left\lceil \frac{k+p}{p+1} \right\rceil$$

Temos que:

$$\left\lceil \frac{k+1}{2} \right\rceil > \left\lceil \frac{k+p}{p+1} \right\rceil$$

já que quanto maior p menor $\left\lceil \frac{k}{p+1} \right\rceil$. Logo, $b(T_{11}) > b(T_{1p})$.

Agora vamos provar que $b(T_{11}) \geq b(T_{ip})$, para $i \leq p$ e $i = \{2, ..., n_1 - 1\}$.

Temos que T_{ip} é um caminho, já que, por definição, os vértices que o constituem não possuem folhas adjacentes. Logo, o $diam(T_{ip})$ é o número de vérteces, $n(T_{ip})$, menos 1. Temos:

$$b(T_{ip}) = \left\lceil \frac{n(T_{ip}) - 1}{n(T_{ip}) - 1} \right\rceil = 1.$$

Logo, $b(T_{11}) \geq b(T_{ip})$.

Agora vamos analizar os casos que incluem o vértice n_1, que também possui folhas adjacentes. Assim como em T_{11}, o número de vértices de de $T_{n_1 n_1}$ é igual ao número de folhas, l mais 2.

$$b(T_{n_1 n_1}) = \left\lceil \frac{n(T_{n_1 n_1}) - 1}{2} \right\rceil = \left\lceil \frac{l + 2 - 1}{2} \right\rceil = \left\lceil \frac{l+1}{2} \right\rceil.$$

Temos que:

- se $k = l$ então $b(T_{11}) = b(T_{n_1 n_1})$;
- se $k < l$ então $b(T_{11}) < b(T_{n_1 n_1})$;
- se $k > l$ então $b(T_{11}) > b(T_{n_1 n_1})$.

Para $b(T_{1n_1})$ temos que:

$$b(T_{1n_1}) = \left\lceil \frac{k+l+n_1-1}{n_1+1} \right\rceil .$$

Vamos provar por indução em n_1 que:

$$\frac{\max(l,k)+1}{2} > \frac{k+l+n_1-1}{n_1+1}$$

com $n_1 \geq 3$, $k, l \geq 2$ e $k \neq l$.

Para $n_1 = 3$, temos que:

$$\frac{k+l+n_1-1}{n_1+1} = \frac{k+l+2}{4} = \frac{(\frac{k+l}{2})+1}{2} < \frac{(2\max(l,k))+1}{2} = \frac{\max(l,k)+1}{2}.$$

Suponha que a desigualdade dada é válida para $n_1 = W$. Vamos provar que a desigualdade também é válida para $n_1 = W+1$. Como a desigualdade é válida para $n_1 = W$, temos que:

$$\frac{k+l+W-1}{W+1} < \frac{\max(l,k)+1}{2}$$

como $W+1 > 0$

$$k+l+W-1 < (W+1)(\frac{\max(l,k)+1}{2})$$

$$(k+l+W-1) + \frac{\max(l,k)+1}{2} < (W+2)(\frac{\max(l,k)+1}{2})$$

$$(k+l+W-1)+1 < (k+l+W-1) + \frac{\max(l,k)+1}{2} < (W+2)(\frac{\max(l,k)+1}{2})$$

$$\frac{k+l+W}{W+2} < (\frac{\max(l,k)+1}{2}).$$

Cabe observar que: $n \geq 3$; como $k-1 > 0$

$$n_1(k-1) \geq 3(k-1)$$

$$n_1 k - n_1 \geq 3k - 3$$

$$n_1k - 2n_1 + n_1 \geq 4k - k - 2 - 1$$

$$n_1k + n_1 + k + 1 \geq 4k + 2n_1 - 2$$

$$(n_1 + 1)(k + 1) \geq 2(2k + n_1 - 1)$$

$$\frac{(k+1)}{2} \geq \frac{2k + n_1 - 1}{n_1 + 1}.$$

Temos desigualdade estrita para $n_1 > 3$ e igualdade para $n_1 = 3$. Assim, quando $k \neq l$, vemos que a desigualdade é satisfeita.

Logo,

- se $k = l$ então $b(T_{11}) = b(T_{n_1n_1}) > b(T_{1n_1})$

$$\left\lceil \frac{l+1}{2} \right\rceil \geq \left\lceil \frac{2l + n_1 - 1}{n_1 + 1} \right\rceil;$$

- se $k < l$ então $b(T_{n_1n_1}) > b(T_{1n_1})$

$$\left\lceil \frac{l+1}{2} \right\rceil > \left\lceil \frac{l + k + n_1 - 1}{n_1 + 1} \right\rceil;$$

- se $k > l$ então $b(T_{11}) > b(T_{1n_1})$

$$\left\lceil \frac{k+1}{2} \right\rceil > \left\lceil \frac{l + k + n_1 - 1}{n_1 + 1} \right\rceil.$$

Todas as outras combinações possíveis possuem demonstração análoga.

□

Essa informação é pertinente, pois assim já não precisamos percorrer todas subárvores do caterpillar, basta saber qual das estrelas possue o maior número de

vértices. A determinação do MINBW($C_{k,l}$) é constante, o que nos permite concluir que a complexidade do algoritmo passa a ser $O(n)$.

Para determinação do layout ótimo, adaptamos o algoritmo mostrado na seção anterior, bastando calcular as duas equações para as estrelas T_1 e T_{n_1}. Para os rótulos de v_2 até v_{n_1-1} utilizamos em ordem os inteiros do intervalo $[1, n]$ que ainda não tenham sido utilizados.

Algoritmo *Largura_de_Banda*;
Entrada: Caterpillar duas estrelas com $n_1 > 2$;
Saída: Layout ótimo do caterpillar;
início
  Seja w_{11} uma folha ligada a v_1, Seja $v_0 = w_{11}$. $L(v_0) \leftarrow 1$;
  $maior_L \leftarrow 1$, $BW \leftarrow 0$;
  $num\_elem(F(v_1)) \leftarrow num\_elem(F(v_1)) - 1$;
  Seja n o número de vértices da maior subárvore T_{ii};
  $BW(L^*, G) \leftarrow \lceil \frac{n-1}{2} \rceil$;
  para $i \leftarrow 1, \ldots, n_1$ **faça**
    se $i = 1$ **ou** $i = n_1$ **então**
      $L_1 = L(v_{i-1}) + BW$;
      $L_2 = maior_L + num\_elem(F(v_i)) + 1$;
      se $L_1 < L_2$ **então**
        $L(v_i) \leftarrow L_1$;
      se não
        $L(v_i) \leftarrow L_2$;
      Numerar todas $w_{ij} \in v_i$ com os menores inteiros não usados do intervalo $[1, n]$;
      $maior_L \leftarrow$ maior $L(w_{ij})$;
    se não
      $L(v_i) \leftarrow maior_L + 1$;
      $maior_L \leftarrow L(v_i)$;
fim.

A Figura 5.5 é o layout ótimo do caterpillar da Figura 5.4, esse caterpillar possui MINBW$(G) = 4$.

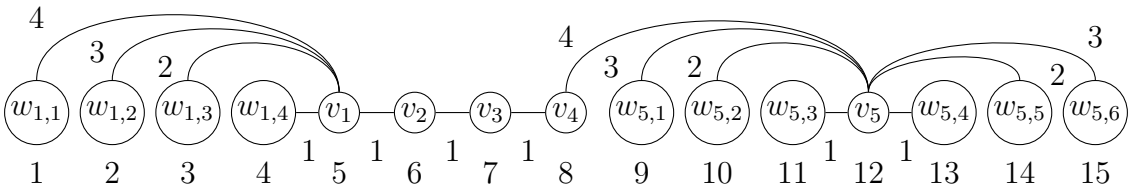

Figura 5.5: Layout ótimo do caterpillar 2-estrelas com $n_1 \geq 3$.

Quando $n_1 = 2$ a solução é obtida diretamente pelo Teorema 5.1.

No próximo capítulo falaremos sobre o problema de determinação do perfil.

6 PERFIL

6.1 Introdução

A determinação do perfil de um grafo foi originalmente proposta como uma abordagem para reduzir os requisitos de espaço para armazenamento de matrizes esparsas [54], sendo também usado para melhorar o desempenho das operações em sistemas de equações não lineares. Neste contexto, o problema é equivalente ao problema *SumCut* [4], outro problema de layout em grafos.

Seja $PR(L,G) = \sum_{u \in V} \left(L(u) - \min_{v \in N[u]} L(v) \right)$. Dado um grafo $G = (V,E)$, o problema de *determinação do perfil* de G consiste em encontrar um layout L^* tal que

$$PR(L^*,G) = \min_{L \in \mathcal{L}} PR(L,G),$$

sendo $\mathcal{L}$ o conjunto de todos os layouts de G. O valor $PR(L^*,G)$ é denotado $\text{MINPR}(G)$,

Os problemas *SumCut* e *perfil* foram definidos separadamente. O primeiro foi definido por Días *et al.* [13] em 1991, já o segundo problema foi definido em 1994 por Lin e Yuan em [57]. Os dois são abordagens equivalentes para o problema *interval graph completion* (complemento de grafo de intervalo) [4] que consiste em encontrar um supergrafo de um grafo G com o menor número de arestas possível. Do ponto de vista algorítmico, o problema perfil é conhecido por ser NP- completo [18]. No entanto, classes especiais de grafos podem ser resolvidos de forma otimizada

em tempo polinomial. Lin e Yuan [57] propuseram vários algoritmos polinomiais para encontrar a solução ótima do perfil para caminhos, rodas, grafos bipartidos completos e árvores com diâmetro 4.

Tratando especificamente de árvores, Kuo e Chang [32] apresentaram um algoritmo com complexidade de tempo $O(n^{1.722})$ para resolver o problema. Esse artigo trata a existência do que é chamado um *caminho básico* na árvore, que possui os centroides da árvore obtidas com a remoção do vértice. Para determinar o centroide é necessário encontrar primeiramente o peso dos vértices. O *peso* de um vértice é o maior número de arestas em uma de suas subárvores. O *centroide* de uma árvore é o vértice que possui o menor peso. No algoritmo, primeiramente é encontrado o centroide da árvore, e em seguida é determinado o caminho básico. O algoritmo é executado recursivamente para as subárvores da árvore.

Para o perfil dos caterpillars não utilizamos o algoritmo apresentado em [32] e sim uma solução que explora a estrutura particular do grafo.

6.2 Perfil para Caterpillars

Nesta seção será provado o valor do perfil e a forma de montar o layout ótimo para a classe dos caterpillars. Primeiramente provaremos a existência de um limite inferior para o valor do perfil de grafos quaisquer, esse resultado será utilizado na determinação ótima do perfil para caterpillar.

Teorema 6.1. *Seja G um grafo conexo e L um layout de G. Então*

$$PR(L, G) \geq m.$$

Prova. Seja $L = \{(v_1, 1), (v_2, 2), ..., (v_n, n)\}$ um layout de G. Sejam: $N^-(v_i) =$

$\{w \in N[v_i] | L(w) < i\}$ e $N^+(v_i) = \{w \in N[v_i] | L(w) > i\}$. Considerando o perfil L, $PR(L, G)$ resulta do somatório de n parcelas $p_1, p_2, ..., p_n$ tais que:

$$p_i = L(v_i) - \min_{w \in N[v_i]} L(w).$$

Como $v_i \in N[v_i]$, podemos considerar inicialmente $p_i \geq 0$. Ao determinamos a parcela p_i, só serão então considerados os vértices $w \in N^-(v_i)$, isto é, somente os w com posições inferiores a v_i em L. E mais, quanto maior a cardinalidade de $N^-(v_i)$, menor é a posição em que se encontra w com $L(w)$ mínimo. Então, $p_i \geq |N^-(v_i)|$.

Logo,

$$PR(L, G) \geq 0 + p_2 + p_3 + ... + p_n$$

$$PR(L, G) \geq 0 + |N^-(v_2)| + |N^-(v_3)| + ...|N^-(v_n)|.$$

Sabemos que $\sum_{v_i \in V} N(v_i) = 2m$. Como cada aresta vw aparece em $N(v)$ e $N(w)$, e no cálculo do perfil só aparece uma vez (em $N^-(v)$) tem-se que $PR(L, G) \geq m$. □

Os caterpillars serão aqui tratados de forma diferente. Será construído um layout particular para os mesmos, que denotaremos L', da seguinte forma: primeiramente posicionamos os vértices da espinha dorsal, v_i, na ordem em que eles aparecem no caterpillar; em seguida posicionamos as folhas adjacentes a v_i à sua esquerda. O layout L' será chamado *layout pré-folhas*. A Figura 6.1 mostra um caterpillar e o layout L'.

O valor de MINPR(G) para um caterpillar é dado no teorema a seguir.

Teorema 6.2. *Seja G um caterpillar. Então* MINPR$(G) = n - 1$.

Prova. Seja G um caterpillar e L' seu layout pré-folhas.

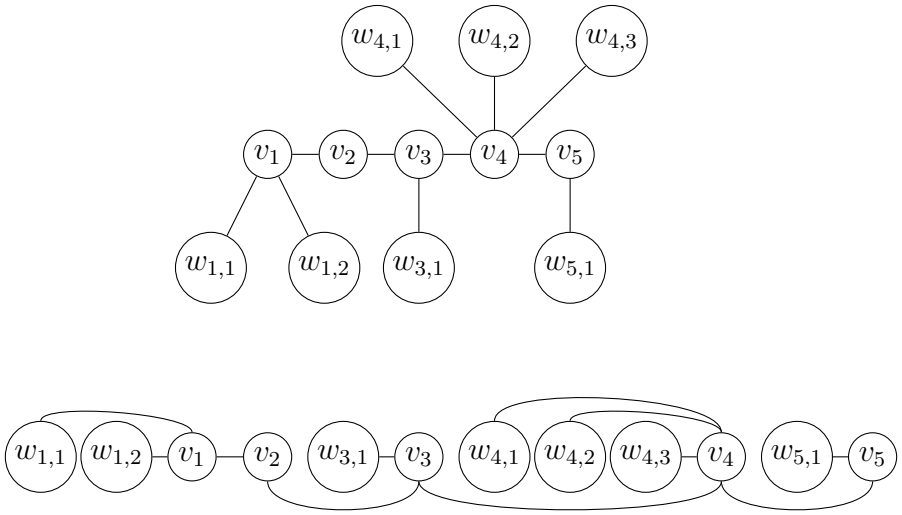

Figura 6.1: Layout L' do grafo G.

Analisando as folhas:

Por construção, as folhas adjacentes a v_i estão à sua esquerda em L'. Então $N[w_{i,j}] = \{w_{i,j}\}$. Logo,

$$L(w_{ij}) - L(w_{ij}) = 0.$$

Analisando os vértices da espinha dorsal:

Para os vértices da espinha dorsal temos:

$$L(u) - \min_{v \in N[u]} L(v) = \begin{cases} L(v_1) - 1, & \text{para} \quad i = 1, \\ L(v_i) - L(v_{i-1}), & \text{para} \quad i = 2, 3, ..., n_1. \end{cases}$$

Logo,

$$\sum_{u \in V} \left(L(u) - \min_{v \in N[u]} L(v) \right) = L(v_1) - 1 + \sum_{i=2}^{i=n_1} L(v_i) - L(v_{i-1}).$$

Observe que $L(v_1) - 1$ é igual ao número de folhas adjacentes à v_1. E $L(v_i) - L(v_{i-1})$ é o número de folhas adjacentes a v_i mais 1. Logo teremos o somatório do

número de folhas adjacentes a cada um dos vértices da espinha dorsal, que é igual a n_2. A partir de v_2 adicionamos 1 ao número de folhas adjacentes, com $n_1 - 1$ parcelas (índice de 2 até n_1). Temos então:

$$\sum_{u \in V} \left(L(u) - \min_{v \in N[u]} L(v) \right) = n_2 + n_1 - 1 = n - 1.$$

Logo, $PR(L', G) = n - 1$ e pelo Teorema 6.1 este valor é mínimo; o layout L' é um layout de custo ótimo. □

O Teorema 6.2 mostra que o layout pré-folhas é ótimo. É imediato concluir que a construção deste layout é simples e tem complexidade de tempo linear.

7 CONCLUSÃO

Neste trabalho foram estudados quatro problemas de layout em grafos: *separação de vértices*, *arranjo linear mínimo*, *largura de banda* e *perfil*; para cada um deles abordamos classes com resultados polinomiais conhecidos, com ênfase nos resultados para a classe das árvores.

As principais contribuições deste trabalho são os resultados obtidos para a classe dos caterpillars. Para a separação de vértices foi visto no Teorema 3.2 que a separação de vértices para carterpillars é constante:

$$\textsc{minvs}(G) = 1,$$

também mostramos o layout ótimo para essa classe.

Para o arranjo linear mínimo, mostramos como encontrar o $\textsc{minla}(G)$ para um caminho, Teorema 4.4, e uma estrela, Corolário 4.1, que são tipos especiais de caterpillars:

arranjo linear mínimo de um caminho:

$$\textsc{minla}(L, G) = |E| = |V| - 1.$$

arranjo linear mínimo de um grafo estrela:

$$\textsc{minla}(G) = \begin{cases} \dfrac{n^2 - 1}{4}, & \text{para} \quad n \quad \text{ímpar}, \\ \dfrac{n^2}{4}, & \text{para} \quad n \quad \text{par}. \end{cases}$$

Além disso, apresentamos o algoritmo dado por Horton em [24] que monta o layout ótimo para o arranjo linear mínimo de um caterpillar dado. A soma das arestas desse layout ótimo é o valor do MINLA(G), sendo G um caterpillar.

No Capítulo 5 foi abordado o problema da largura de banda. Apresentamos o Teorema 5.1, dado por Syslo e Zack em [52], que determina o valor da largura de banda para um caterpillar. Também definimos um caterpillar especial, que chamamos de *caterpillar 2-estrelas*. Esse caterpillar possui resultado diferente quando: $n_1 \geq 3$; provamos no Teorema 5.2 esse resultado. Para o layout ótimo da largura de banda para os caterpillars 2-estrelas nós adaptamos o algoritmo dado por Syslo e Zack em [52], alterando o cálculo da largura de banda. Essa mudança otimiza o algoritmo pois assim ele não necessita passar por todos os subgrafos contidos no caterpillar.

O perfil de um caterpillar foi tratado no Capítulo 6. Primeiramente foi provado que dado um grafo G conexo, ele possui o seginte limite inferior:

$$PR(L, G) \geq m.$$

Esse resultado foi visto no Teorema 6.1 Para prova do Teorema 6.2 foi usado o resultado visto logo a cima. Esse teorema mostra que o valor para o perfil de um caterpillar G é:

$$\text{MIMPR}(G) = n - 1.$$

Devido aos bons resultados obtidos para os grafos caterpillar pretendemos continuar nossos estudos para essa classe de grafos em outros problemas de layout.

REFERÊNCIAS

[1] ADOLPHSON, D. Single machine job sequencing with precedence constraints. **SIAM Journal on Computing**, v. 6, n. 1, p. 40-54, 1977.

[2] ADOLPHSON, D. e HU, T. C. Optimal linear ordering. **SIAM Journal on Applied Mathematics**, v. 25, n. 3, p. 403-423, 1973.

[3] ASSMAN, S. F.; PECK, G. W.; SYSLO, M.M.; ZAK, J. The Bandwidth of Caterpillars With Hair of Lengths 1 and 2 . **SIAM Journal Algebraic and Discrete Methods**, v. 2, n. 4, p. 387-393, 1981.

[4] AGRAWAL, A. P.; KLEIN; RAVI, R. Ordering problems approximated: single-processor scheduling and interval graph completion. **Automata, Languages and Programming**, 510, (1991) 751-762.

[5] BOLLOBÁS, B.; LEADER, I. **Edge– Isoperimetric Inequalities in the Grid. Combinatorica**, v. 11, n. 4, p. 299-314, 1991.

[6] BODLAENDER, H.L.; MOHRING, H. R. The pathwidth and treewidth of cografos. **SIAM Journal on Discrete Mathematics**, v. 6, n. 2, p. 181-190, 1990.

[7] BOTAFOGO, R.A. Cluster analysis for hypertext systems. In: INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 16., 1993, Pittsburgh. **Proceedings** 1993, p. 116-125.

[8] CHINN, P. Z.; CHVÁTALOVÁ, J.; DEWDNEY, A. K.; GIBBS, N. E. The bandwidth problem for graphs and matrices a survey. **Journal of Graph Theory**, v. 6, n. 3, p. 223-254, 1982.

[9] CHOU, Hsin-Hung; KO, Ming-Tata; HO, Chin-Wen; CHEN, Gen-Huey. On the vertex separation of unicyclic graphs. In: WORKSHOP ON COMBINATORIAL MATHEMATICS AND COMPUTATION THEORY, 23., 2006, Taiwan. **Proceedings...** 2006, p. 28-33.

[10] CHUNG, F. R. K. On optimal liear arrangements of tree. **Computers & Mathematics with Applications**, v. 10, n. 1, p. 43-60, 1984.

[11] COHEN, J.; FORMIN, F.; HEGGERNES, P.; KRATSCH, D.; KUCHEROV, G. Optimal linear arrangement of interval graphs. **Mathematical Foundations of Computer Science**, v. 4162, p. 267-279, 2006.

[12] DÍAZ, J.; PETIT, J.; SERNA, M. A survey of graph layout problems. **ACM Computing Surveys**, v. 34, n. 3, p. 313-356, 2002.

[13] DIÁZ, J.; GIBBONS, A. M.; PATERSON, M. S.; TORÁN, J. The minsumcut problem. In: DEHNE, F.; SACK, J. R. (Ed.). **Algorithms and data structures.** New York: Springer-Verlag, 1991. p. 65-79. (Lecture Notes in Computer Science, 519).

[14] DIESTEL, R. **Graph theory.** 2 ed. New York: Springer Verlag, 2000.

[15] ELLIS, A. J.; SUDBOROUGH, I. H.; TURNE, J. S. The vertex separation and search number of a graph. **Information and Computation**, v. 113, n. 1, p. 59-79, 1994.

[16] ELLIS, J.; MARKOV, M. Computing the vertex separation of unicyclic graphs. **Information and Computation**, v. 192, n. 2, p. 123-161, 2004.

[17] FALLAT, S.; KIRKLAND, S. Extremizing algebraic connectivity subject to graph theoretic constrains. **The Electronic Journal of Linear Algebra**, v. 3, p. 48-74, 1998.

[18] GAREY, M. R; JOHNSON, S. D. **Computers and Intractability**: a guide to the theory of NP-Completeness. San Francisco: W. H. Freeman, 1979.

[19] GOLDBERG, K. M.; KLIPKER, A. I. An algorithm for minimal numeration of tree vertices. Sakharth. **SSR Mecn. Acad. Moambe**, v. 81, p. 553-556, 1976.

[20] HARARY, F.; SCHWENK, A. J. The number of caterpillars. **Discrete Mathematics**, v. 6, n. 4, p. 359-365, 1973.

[21] HARPER, L. H. Optimal assignments of numbers to vertices. **Journal of the Society for Industrial and Applied Mathematics,** v. 12, n. 1, p. 131-135, 1964.

[22] HEGGERNES, P.; KRATSCH, D.; MEISTER, D. Bandwidth of bipartite permutation graphs in polynomial time. In: LABER, E. S. [et al] (Ed.).**LATIN 2008:** theoretical informatics. Berlin: Springer, 2008. p. 216-227. (Lecture Notes in Computer Science, 4957).

[23] HOCHBERG, R. A.; STALLMANN, M. F. Optimal one-page tree embeddings in linear time. **Information Processing Letters**, v. 87, n. 2, p. 59-66, 2003.

[24] HORTON, S. B. **The optimal linear arrangement problem:** algorithms and approximation. Tese. Georgia Institute of Technology, Atlanta, 1997.

[25] KARP, R. M. Mapping the genome: some combinatorial problems arising in molecular biology. In: SYMPOSIUM ON THEORY OF COMPUTING, 23., 1993, San Diego. **Proceedings...** 1993, p. 278-285.

[26] KENDALL R. Incidence matrices, interval graphs and seriation in archaeology. **Pacific Journal of Mathematics,** v. 28, n. 3, p. 565-570, 1969.

[27] KIROUSIS, L. M.; PAPADIMITRIOU, C. H. Interval graphs and searching. **Discrete Mathematics,** v. 55, n. 2, p. 181-184, 1985.

[28] _____. Searching and pebbling. **Theoretical Computer Science,** v. 47, p. 205-218, 1986.

[29] KLEITMAN, D. J.; VOHRA, R. V. Computing the bandwidth of interval graphs. **SIAM Journal on Discrete Mathematics,** v. 3, n. 3, p. 373-375, 1990.

[30] KLOKS, T.; BODLAENDER, H. **On the treewidth and pathwidth of permutation graphs.** Utrecht: Department of Computer Science, 1992. (Technical Report RUU-CS- 92-13). .

[31] KLOKS, T.; KRATSCH, D.; MULLER H. Bandwidth of chain graphs. **Information Processing Letters,** v. 68, n. 6, p. 313-315, 1998.

[32] KUO, D.; CHANG, G. J. The prole minimization problem in trees. **SIAM Journal on Computing,** v. 23, n. 1, p. 71-81, 1994.

[33] LAI, Yung-Ling.; WILLIANS, K. A survey of solved problems and applications on bandwidth, edgesum, and prole of graphs. **Journal of Graph Theory,** v. 31, n. 2, p. 75-94, 1999.

[34] LEISERSON, C. E. Area-eficient graph layout. In: SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, 21., 1980, Syracuse. **Proceedings...** 1980, p. 270-281.

[35] MEGIDDO, N.; HAKIMI, S. L.; GAREY, M. R.; JOHNSON, D. S.; PAPADIMITRIOU, C. H. The complexity of searching a graph. **Journal of the ACM,** v. 35, n.1, p. 18-44, 1988.

[36] MONIEN, B. The bandwidth minimization problem for caterpillars with hair length 3 is NP-Complete. **SIAM Journal on Algebraic Discrete Methods,** v. 7, n. 4, p. 505-512, 1986.

[37] MITCHISON, G; DURBIN, R. Optimal numberings of an $N \times N$ array. **SIAM Journal on Algebraic Discrete Methods,** v. 7, n. 4, p. 571-582, 1986.

[38] OLIVEIRA, S. K. K.; JUSTEL, C. M. Determinação de vertex separation para árvores binárias cheias. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 37., 2005, Gramado. **Anais...** 2005, p. 2343-2352.

[39] PARSONS, T. D. Pursuit-evasion in a graph. In: ALAVI, Y.; LICK, D. R. **Theory and Application of Graphs.** Berlin: Springer, 1976. p. 426-441. (Lecture Notes in Mathematics, 426).

[40] _____. The search number of a connected graph. In: SOUTHEASTERN CONFERENCE ON COMBINATORICS, GRAPH THEORY, AND COMPUTING, 9., 1978, Florida. **Proceedings...** 1978, p. 549-554.

[41] PAPADIMITRIOU, C. H. The NP-completeness of the bandwidth minimization problem. **Computing,** v. 16, n. 3, p. 263-270, 1976.

[42] PETIT, J. Addenda to the survey of layout problems. **Bulletin of the EATCS,** n. 105, p. 177-201, 2011.

[43] RAVI, R.; AGRAWAL, A. P.; KLEIN, P. Ordering problems approximated: single-processor scheduling and interval graph completion. In: ALBERT, J. L.; MONIEN, B.; RODRÍGUEZ-ARTALEJO, M. (Ed.). **Automata, Languages and Programming.** Berlin: Springer, 1991. p. 751-762. (Lecture Notes in Computer Science, 510).

[44] ROBERTSON, N.; SEYMOUR, P. D. Graph minors. I. Excluding a forest. **Journal of Combinatorial Theory, Series B,** v. 35, n. 1, p. 39-61, 1983.

[45] _____. Disjoint paths - a survey. **SIAM Journal on Algebraic Discrete Methods,** v. 6, n. 2, p. 300-305, 1985.

[46] SAAD, Y. **Iterative methods for sparse linear systems.** 2. ed. Philadelphia : SIAM, 2003.

[47] SANTA RITA, Vitor Augusto Ferreira. **Bandwidth em grafos.** Dissertação (Mestrado) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2010.

[48] SHAHROKHI, F.; SÝKORA, O.; SZÉKELY, L. A.; VRTO I. On bipartite drawing and the linear arrangement problem. **SIAM Journal on Computing,** v. 30, n. 6, p. 1773-1789, 2001.

[49] SHILOACH, Y. A minimum linear arrangement algorithm for undirected trees. **SIAM Journal on Computing,** v. 8, n. 1, p. 15-32, 1979.

[50] SKODINIS, K. Computing optimal linear layouts of trees in linear time. In: Paterson, M. S. **Algorithms:** ESA 2000. Berlin: Springer, 2000. p. 403-414. (Lecture Notes in Computer Science, 1879).

[51] SKODINIS, K. Construction of linear tree-Layouts which are optimal with respect to vertex separation in linear time. **Journal of Algorithms,** v. 47, n. 1, p. 40-59, 2003.

[52] SPRAGUE P. A. An O(n log n) algorithm for bandwidth of interval graphs. **SIAM Journal on Discrete Mathematics,** v. 7, n. 2, p. 213-220, 1994.

[53] SYSLO, M. M. ; ZAK, J. The bandwidth problem: critical subgraphs and the solution for caterpillars. **Annals of Discrete Mathematics,** v. 16, p. 281-286, 1982.

[54] TEWARSON, R. P. **Sparse matrices.** New York: Academic Press, 1973.

[55] UNGER, W. The complexity of the approximation of the bandwidth problem. In: SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, 39., 1998, Palo Alto. **Proceedings...** 1998, p. 82-91.

[56] YAN, J. H. The bandwidth problem in cographs.**Tamsui Oxford Journal of Mathematical Sciences.** v. 13, p. 31-36, 1997.

[57] YUXIN, Lin; JINJIANG, YUAN. Prole minimization problem for matrices and graphs. **Acta Mathematicae Applicatae Sinica,** v. 10, n. 1, p, 107-112, 1994.

[58] ZABUDSKIÃ, G. G. Algorithm for solving a problem on optimal linear ordering. **Russian Mathematics** (Iz. VUZ). v. 41, n. 12, p. 73-78, 1997.