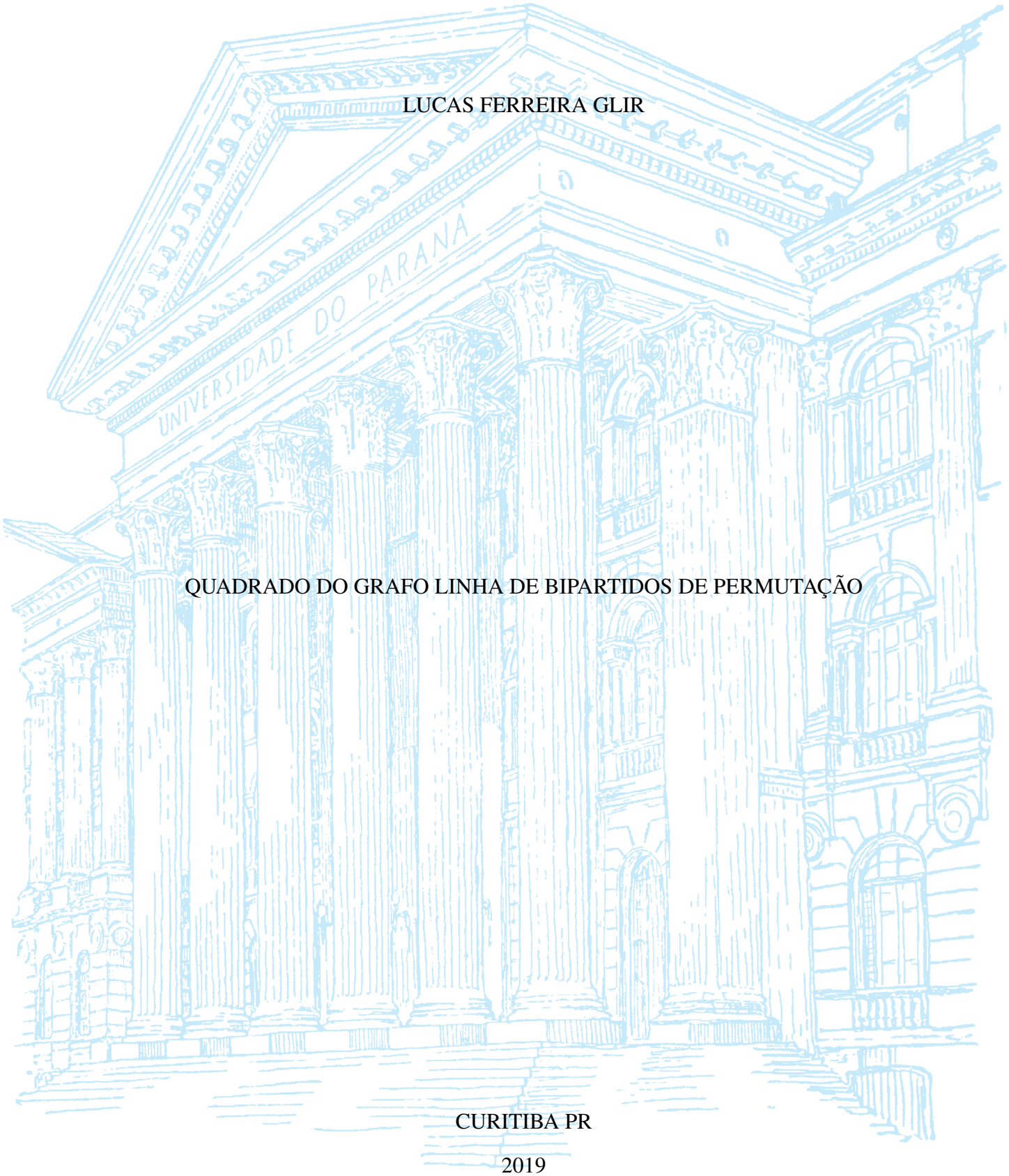
UNIVERSIDADE FEDERAL DO PARANÁ

LUCAS FERREIRA GLIR



QUADRADO DO GRAFO LINHA DE BIPARTIDOS DE PERMUTAÇÃO

CURITIBA PR

2019

LUCAS FERREIRA GLIR

QUADRADO DO GRAFO LINHA DE BIPARTIDOS DE PERMUTAÇÃO

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: André Luiz Pires Guedes.

CURITIBA PR

2019

A Diocesa Ferreira da Silva e Celina Ferreira da Silva, minhas tias, que sempre me apoiaram e não tiveram a oportunidade de me ver completar um dos meus sonhos.

Agradeço, principalmente, a meus pais e meu irmão, pelo apoio e incentivo. Todas as vezes que pensei em desistir ou em que a situação estava difícil, eles estavam sempre presentes para me mostrar que com dedicação e foco posso alcançar meus objetivos.

Agradeço a André Luiz Pires Guedes pela paciência e dedicação ao me orientar.

Agradeço, também, a todos que direta ou indiretamente fizeram parte da minha formação.

RESUMO

Neste trabalho é mostrado o problema de decidir se um grafo H dado é grafo de intersecção de bicliques de algum grafo bipartido G. Quando G é restringido aos bipartidos de permutação, esse problema é equivalente a decidir se o grafo H dado é o grafo quadrado do grafo linha do grafo simplificado de G. Ou seja, $H \approx KB(G) \approx (L(S(G))^2$. O caminho seguido foi o de encontrar o grafo G a partir do grafo H. Portanto, é necessário encontrar as raízes quadradas de H, a pré-imagem de grafo linha dessas raízes e verificar se essas pré-imagens são grafos bipartidos de permutação.

Palavras-chave: Grafo Quadrado. Grafo Bipartido de Permutação. Grafo Linha. Grafo De Intersecção De Bicliques.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Um grafo biclique é o grafo de intersecção das bicliques de um grafo G. Cada vértice é uma biclique de G. Dois vértices em um grafo biclique de G têm uma aresta os interligando se as bicliques de G, representadas por esses vértices, possuem vértices em comum. Essa definição está presente em (Cruz et al., 2018).

Dado um grafo H, queremos saber se $H \approx KB(G)$, para algum G. $KB(G)$ denota o grafo biclique de algum grafo G. Um caminho bastante intuitivo para solucionar esse problema é tentar encontrar uma pré-imagem de grafo biclique de H. Todavia, essa operação é difícil de computar. Apesar de haver algoritmos que resolvem o problema, o espaço de busca é muito extenso e a busca é exaustiva. Quando restringimos G aos grafos bipartidos de permutação, há uma outra forma de encontrá-lo.

Um grafo G é um grafo bipartido de permutação quando ele é, ao mesmo tempo, um grafo bipartido e um grafo de permutação. Como provado em (Spinrad et al., 1987), um grafo bipartido de permutação respeita duas propriedades: a *propriedade da adjacência* e a *propriedade do enclausuramento*.

Se G for um grafo bipartido de permutação, o problema de decidir se $H \approx KB(G)$ é equivalente a decidir se $H \approx (L(S(G)))^2$, como provado em (Cruz et al., 2018). Também há, em (Groshaus e Guedes, 2018), outra equivalência, que não será abordada neste trabalho: $H \approx KB_m(G)$. $KB_m(G)$ denota o grafo biclique de bicliques mutuamente incluídas. A figura 1.1 ilustra essas equivalências.

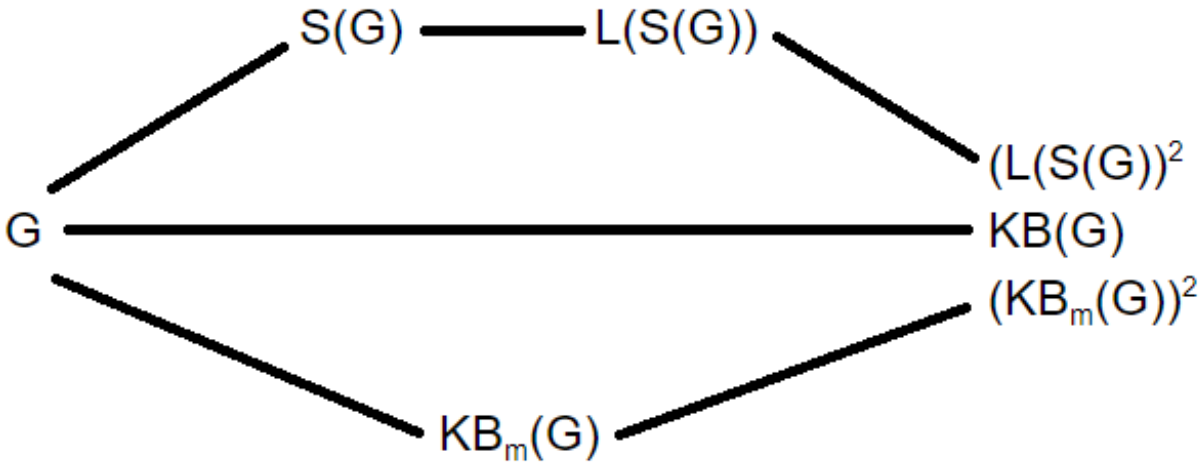


Figura 1.1: esquema das relações.

Um grafo simplificado é o grafo construído através das arestas principais de um grafo bipartido de permutação G. $S(G)$ denota o grafo simplificado de G. A definição deste tipo de grafo está presente em (Cruz et al., 2018).

Um grafo linha é o grafo de intersecção das arestas de um grafo G. $L(G)$ denota o grafo linha de G. Um grafo G qualquer é elevado ao quadrado quando se adiciona a aresta (x, y) entre dois vértices x e y com $d(x, y) = 2$. G^2 denota que G foi elevado ao quadrado.

Encontrar a pré-imagem de $KB(G)$, como dito anteriormente, é bastante difícil, mesmo G sendo bipartido de permutação. Todavia, restringindo G aos bipartidos permutação, podemos tentar encontrar G a partir de $H \approx (L(S(G)))^2$. Portanto, encontramos uma raiz quadrada de

$(L(S(G)))^2$, encontrar a pré-imagem de grafo linha dessa raiz e verificar se essa pré-imagem é bipartida de permutação. Caso ela não seja bipartida de permutação, encontramos outra raiz. Se for, encontramos a pré-imagem de grafo simplificado, que é o G buscado.

Neste trabalho, é apresentado o problema de encontrar algum grafos F, com $F \approx S(G)$, dado um grafo H, tal que $H \approx (L(F))^2$.

2 DEFINIÇÕES

2.1 GRAFO

Um grafo $G = (V, E)$ é um conjunto de vértices $V = \{v_1, v_2, ...\}$ e junto com um conjunto de arestas $E = \{e_1, e_2, ...\}$ que conectam pares de vértices. Arestas podem ser vistas como tuplas de vértices, tal qual $e_m = (v_x, v_y)$. Um grafo cujo conjunto V possui apenas um membro é chamado de grafo trivial. A figura 2.1 descreve, a esquerda e a direita respectivamente, um grafo trivial e um grafo com 5 vértices e 5 arestas.

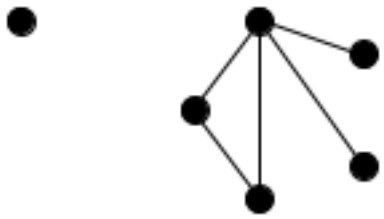


Figura 2.1: grafo trivial e grafo com 5 vértices e 5 arestas.

2.2 VIZINHANÇA DE GRAFO

Dois vértices x e y são vizinhos, ou adjacentes, se há uma aresta (x, y) dentro de um grafo G. Uma vizinhança é o conjunto de vizinhos de um vértice x. $N(x)$ denota a vizinhança de um vértice x. A figura 2.2 descreve um grafo, cujo conjunto $V = \{1, 2, 3, 4, 5, 6\}$ e conjunto $E = \{ (1, 2), (1, 3), (1, 4), (2, 5), (3, 4), (3, 5), (3, 6), (4, 6), (5, 6)\}$. A vizinhança do vértice 3 é composta pelos vértices 1,4,5 e 6.

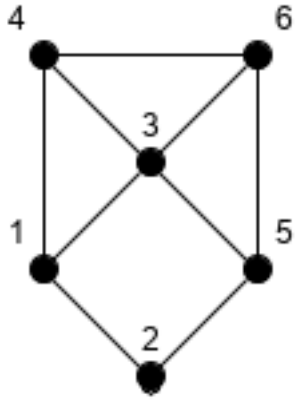


Figura 2.2: grafo exemplo para vizinhança.

2.3 CAMINHO EM GRAFO

Um caminho é uma sequência de vértices $(x_1, x_2, x_3, \ldots, x_{n-1}, x_n)$, tal que $\{(x_1, x_2)$, (x_2, x_3), (x_3, x_4), $\ldots$, (x_{n-2}, x_{n-1}), $(x_{n-1}, x_n)\}$ são arestas e que cada vértice é distinto. Uma sequência de vértices $(x_1, x_2, x_3, \ldots, x_{n-1}, x_n)$ denota um caminho. Um caminho mínimo é o menor caminho entre dois vértices em um grafo. Um grafo G é dito conexo se há pelo menos um caminho entre cada par de vértices do grafo G. A figura 2.3 descreve um grafo, cujo conjunto $V = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ e conjunto $E = \{$ $(0, 1)$, $(0, 2)$, $(1, 2)$, $(1, 3)$, $(2, 3)$, $(3, 4)$, $(4, 5)$, $(4, 8)$, $(5, 6)$, $(5, 8)$, $(6, 7)$ $\}$. Em destaque, temos o caminho $(0, 1, 3, 4, 5, 6, 7)$. É importante notarmos que esse caminho também é mínimo, pois não há caminho menor que este para chegar do vértice 0 ao vértice 7.

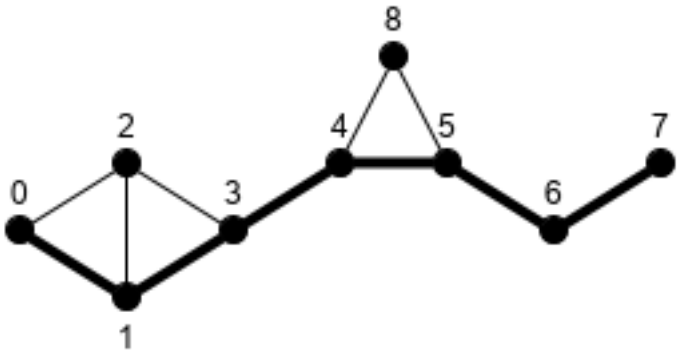


Figura 2.3: exemplo de caminho em grafo.

2.4 CICLO

Um ciclo é um caminho fechado, cujo vértice inicial e vértice final são o mesmo vértice. A cintura de um grafo G é o menor ciclo dentro de um grafo G. No exemlo anterior, na figura 2.3, um dos ciclos existentes no grafo é representado pela sequência $(0, 1, 3, 2, 0)$.

2.5 SUBGRAFO

Um grafo $G_s = (V_s, E_s)$ é subgrafo de $G = (V, E)$ quando $V \supseteq V_s$ e $E \supseteq E_s$. A figura 2.4 descreve um grafo, cujo conjunto $V = \{1, 2, 3, 4, 5, 6\}$ e conjunto $E = \{$ $(1, 2)$, $(1, 3)$, $(1, 4)$, $(2, 5)$, $(3, 4)$, $(3, 5)$, $(3, 6)$, $(4, 6)$, $(5, 6)\}$, e um subgrafo, cujo conjunto $V = \{1, 2, 3, 4, 5, 6\}$ e conjunto $E = \{$ $(1, 2)$, $(1, 3)$, $(1, 4)$, $(2, 5)$, $(3, 4)$, $(3, 5)$, $(3, 6)$, $(4, 6)$, $(5, 6)\}$.

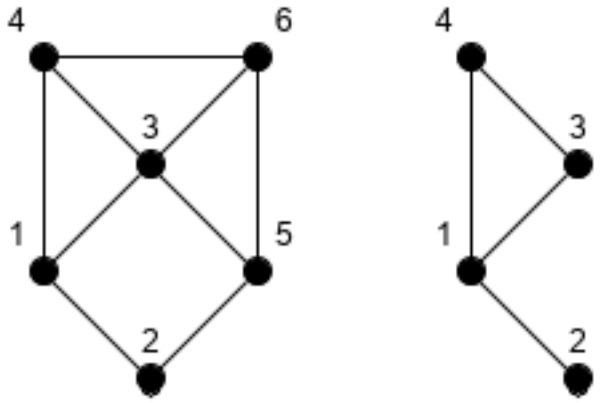


Figura 2.4: exemplo de subgrafo.

2.6 GRAFO BIPARTIDO

Um grafo G é dito bipartido quando os vértices do conjunto V podem ser dividido em dois conjuntos A e B, tais que toda aresta de G une um vértice de A a outro de B. $G(\mathrm{A}, \mathrm{B}, E)$ denota um grafo bipartido com conjuntos A e B. A figura 2.5 descreve um grafo, cujo conjunto, por exemplo, A = $\{0, 1, 2, 3\}$ e conjunto B = $\{4, 5, 6\}$.

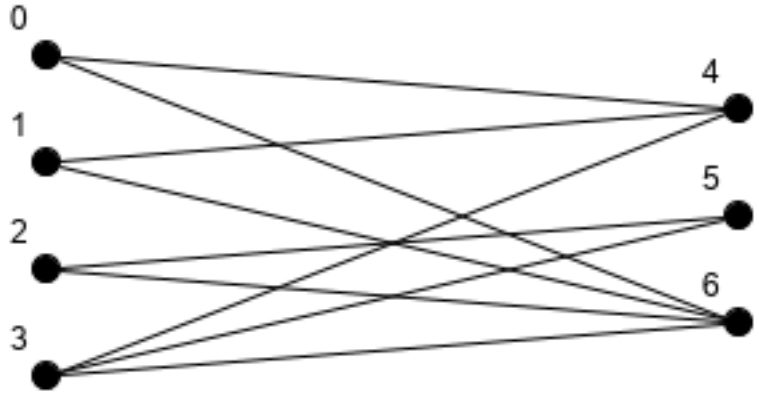


Figura 2.5: exemplo de grafo bipartido.

2.7 GRAFO DE PERMUTAÇÃO

Um grafo G é um grafo de permutação, pela definição dada em (Spinrad et al., 1987), se há um par de permutações P, Q do seu conjunto de vértices de forma que haja uma aresta entre um vértice v e um vértice u se, e somente se, v precede u em uma das permutações $\{P, Q\}$, enquanto u precede v na outra.

Um grafo de permutação definido pelas permutações P, Q pode também ser descrito pelo diagrama de permutação. Consideremos duas colunas, uma constituída dos vértices na ordem dada por P, e a outra constituída dos vértices na ordem dada por Q. Uma linha liga o vértice v na coluna da esquerda ao vértice v na coluna da direita. Há uma aresta conectando os vértices v e u se, e somente se, v precede u em uma coluna e u precede v na outra, ou seja,

há um cruzamento de linhas no diagrama de permutação. Notemos que diversos diagramas de permutação podem representar o mesmo grafo. A figura 2.6 descreve um grafo, cujo conjunto $V = \{0, 1, 2, 3, 4\}$. $\{4, 3, 0, 1, 2\}$ é uma permutação desses vértices. Os cruzamentos entre as linhas no diagrama de permutação, descrito na figura, formam o conjunto $E = \{ (0, 3), (0, 4), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4) \}$.

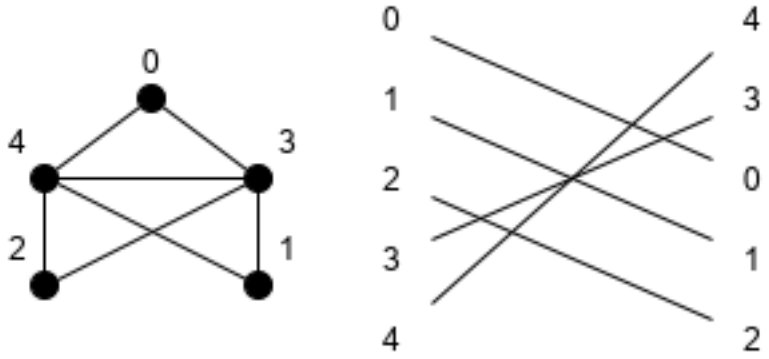


Figura 2.6: exemplo de grafo de permutação e diagrama de permutação.

2.8 GRAFO COMPLETO

Um grafo G é um grafo completo quando cada um de seus pares de vértices estão ligados por uma aresta. K_n denota um grafo completo com n vértices. A figura 2.7 descreve um exemplo de K_6.

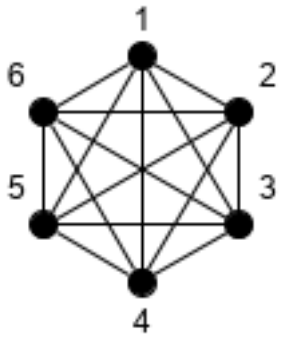


Figura 2.7: exemplo de grafo completo.

2.9 CLIQUE

Uma clique dentro de um grafo G é um subgrafo completo maximal. A figura 2.8 descreve um grafo com 3 cliques, sendo uma clique a aresta com vértices $\{4, 5\}$, uma clique de três vértices, cujos vértices são $\{0, 1, 2\}$, e uma clique de quatro vértices, com os vértices $\{1, 2, 3, 4\}$.

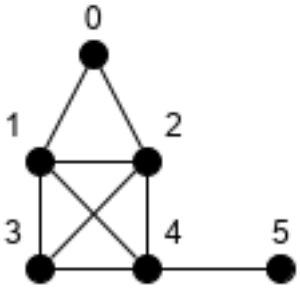


Figura 2.8: exemplo de clique.

2.10 BICLIQUE

Uma biclique dentro de um grafo G é um subgrafo bipartido completo maximal. A figura 2.9 descreve um grafo com três bicliques, cada uma com quatro vértices. Uma é um ciclo com os vértices $\{1, 2, 3, 4\}$, outra com os vértices $\{0, 1, 2, 3\}$ e a terceira com os vértices $\{1, 3, 4, 5\}$.

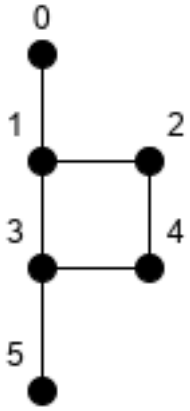


Figura 2.9: exemplo de biclique.

2.11 GRAFOS DE INTERSECÇÃO

Um grafo G é um grafo de intersecção quando ele representa as intersecções entre um ou mais conjuntos. Cada conjunto é representado por um vértice e cada intersecção é representada por uma aresta.

2.12 GRAFO DE INTERSECÇÃO DAS BICLIQUES

Um grafo biclique é grafo de intersecção entre as bicliques de um grafo G. Cada vértice é uma biclique de G. Dois vértices em um grafo biclique de G tem uma aresta interligando-os se, e somente se, as bicliques, representadas por esses vértices, possuem vértices em comum. A figura 2.10 descreve um grafo, a esquerda, com 5 bicliques: R, com A = $\{1\}$ e B = $\{0, 2, 4, 5\}$, S, com A = $\{5\}$ e B = $\{1, 3, 6, 7\}$, T, com A = $\{1, 3\}$ e B = $\{2, 4, 5\}$, U, com A = $\{6\}$ e

B = {5, 10, 11, 12, 13}, e V, com A = {6, 8, 9} e B = {10, 11, 12, 13}. A direita, está descrito o grafo de intersecção dessas bicliques, com conjunto V = { (R, S), (R, T), (R, U), (S, T), (S, U), (S, V), (T, U), (U, V) }.

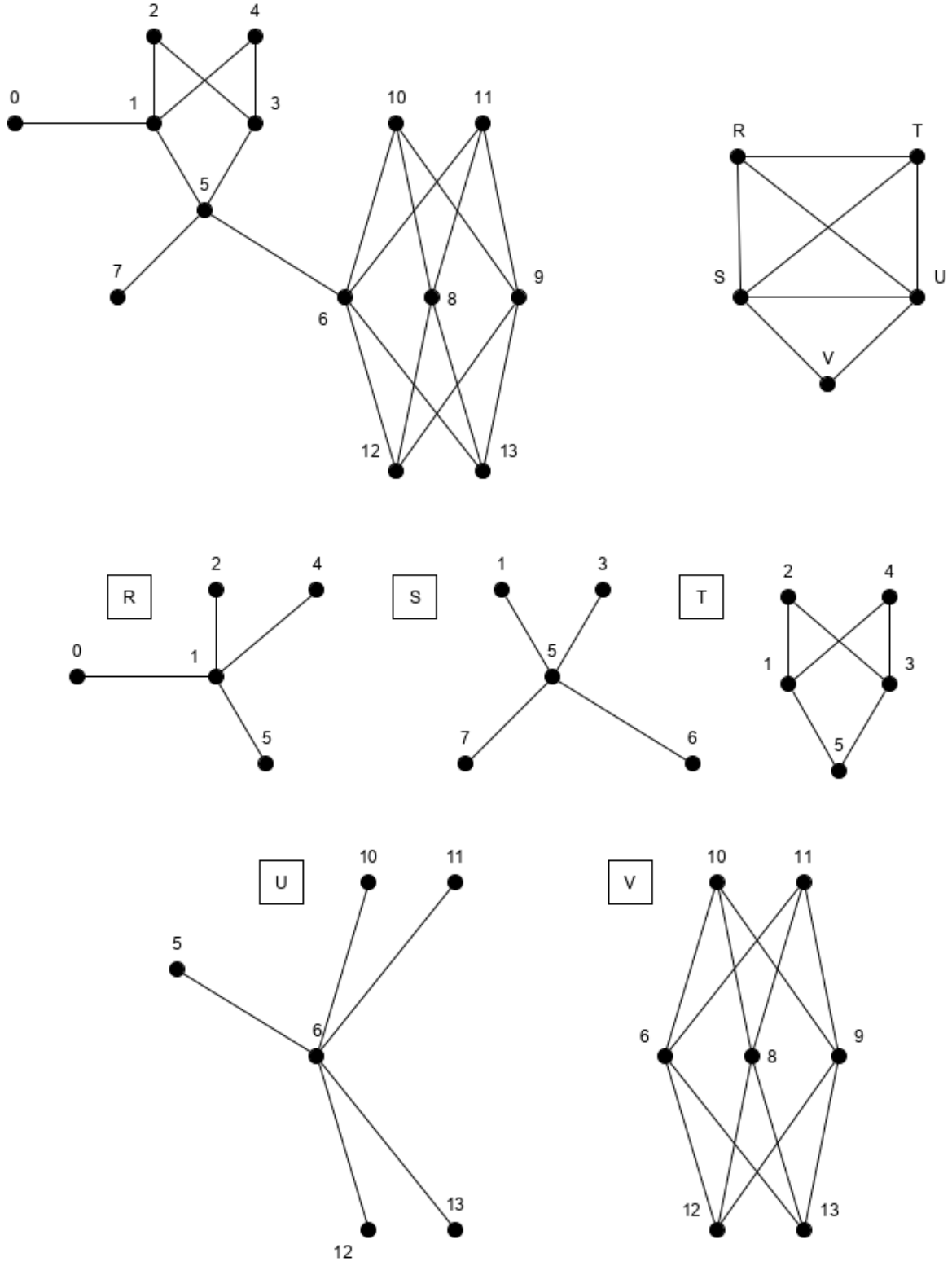


Figura 2.10: exemplo de grafo com bicliques e o grafo de intersecção das bicliques.

2.13 DISTÂNCIA

A distância entre dois vértices x e y em um grafo G é o número de arestas em um caminho mínimo entre eles. $d(x, y)$ denota a distância entre x e y em um grafo G.

2.14 ISOMORFISMO DE GRAFOS

Sejam dois grafos $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$. Dizemos que G_1 e G_2 são isomorfos se existe um mapeamento bijetivo $f : V_1 \iff V_2$ tal que $(x, y) \in E_1$ se, e somente se, $(f(x), f(y)) \in E2$, para todo $x, y \in V_1$. $G_1 \approx G_2$ denota que G_1 é isomorfo a G_2. A figura 2.11 descreve dois grafos isomorfos. A função f, na figura, é uma função bijetora dos vértices do grafo a esquerda, cujo conjunto $V = \{0, 1, 2, 3, 4\}$, para os vértices do grafo a direita, cujo conjunto $V = \{A, B, C, D, E\}$.

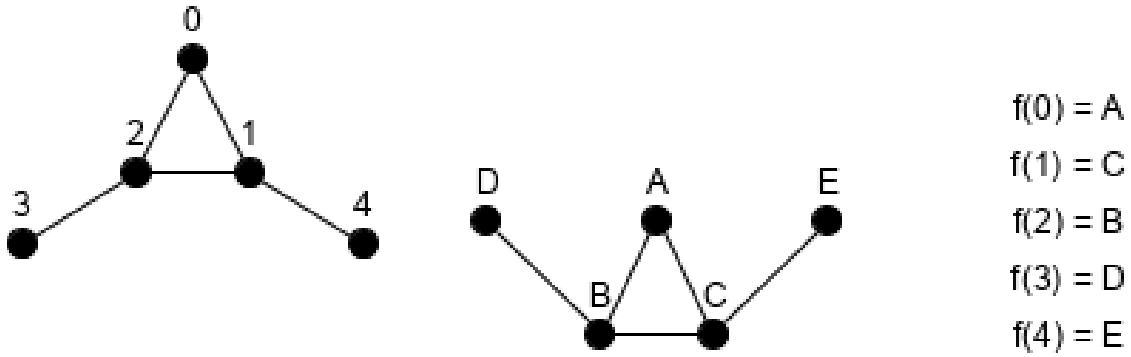


Figura 2.11: exemplo de isomorfismo de dois grafos.

2.15 GRAFO BIPARTIDO DE PERMUTAÇÃO

Um grafo G é um grafo bipartido de permutação quando ele é, ao mesmo tempo, um grafo bipartido e um grafo de permutação. Três definições e um teorema, definidos e provado em (Spinrad et al., 1987), são necessários para explicar o algoritmo de reconhecimento de grafos bipartidos de permutação. Considere que a bipartição dos vértices de G é (A, B).

Definição 2.15.1. Uma ordenação dos vértices de A em um grafo bipartido $G = (V, E)$ tem a *propriedade da adjacência* se para cada vértice x em B, $N(x)$ é constituída de vértices que são consecutivos na ordenação de A.

Definição 2.15.2. Uma ordenação dos vértices A em um grafo bipartido $G = (V, E)$ tem a *propriedade do enclausuramento* se para cada par de vértices x, y em B e tal que $N(x)$ é um subconjunto de $N(y)$, vértices em $N(y) - N(x)$ são consecutivos dentro de A.

Definição 2.15.3. Uma ordenação forte dos vértices de um grafo bipartido $G = (V, E)$ é constituída de uma ordenação de A e uma ordenação de B tal que para todo (α, β) e $(\alpha,, \beta,)$ em E,onde $\alpha, \alpha,$ são vértices de A e $\beta, \beta,$ são vértices de B, $\alpha < \alpha,$ e $\beta > \beta,$ implica que $(\alpha, \beta,)$ e $(\alpha,, \beta)$ existem em E.

Teorema 2.15.1. *As seguintes afirmações são equivalentes para um grafo bipartido $G = (V, E)$:*

i *G é um grafo bipartido de permutação*

ii *Há uma ordenação forte de* AB.

iii *Há uma ordenação de* A *que possui a propriedade da adjacência e a propriedade do enclausuramento.*

A figura 2.12 descreve um grafo bipartido de permutação a esquerda, com $V = \{0, 1, 2, 3, 4\}$, e seu diagrama de permutação a direita, que forma $E = \{ (0, 2), (1, 2), (1, 4), (3, 4) \}$.

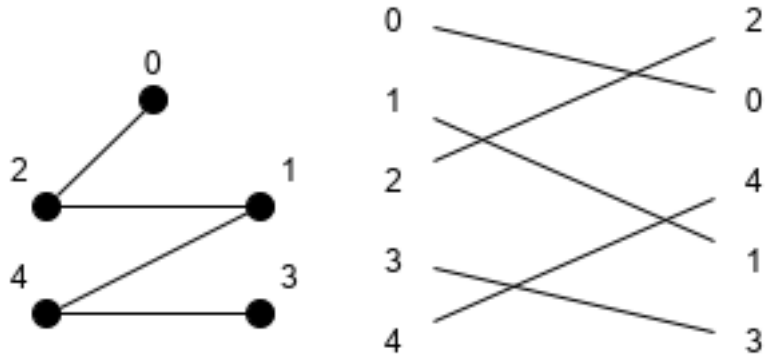


Figura 2.12: exemplo de grafo bipartido de permutação e diagrama de permutação.

2.16 GRAFO LINHA

Um grafo linha é o grafo de intersecção entre as arestas de um grafo G. Cada vértice é uma aresta de G. Dois vértices em um grafo linha de G tem uma aresta interligando-os se, e somente se, as arestas representadas por esses vértices compartilham um mesmo vértice em G. $L(G)$ denota o grafo linha de um grafo G. A figura 2.13 descreve um grafo, com $E = \{ (0, 1), (1, 2), (1, 3), (2, 3) \}$, e seu grafo linha. Os vértices do grafo linha são nomeados de acordo com as arestas que representam.

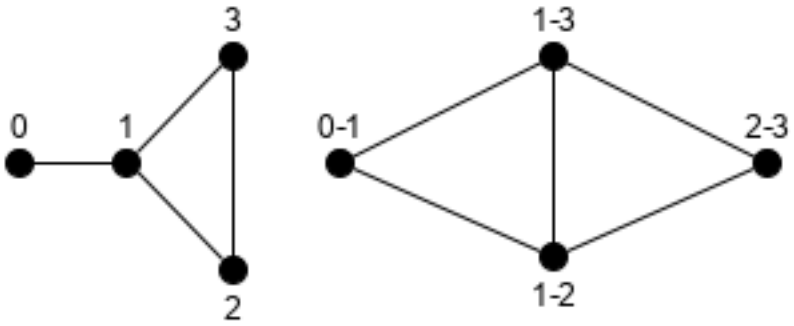


Figura 2.13: exemplo de grafo bipartido de permutação e diagrama de permutação.

2.17 GRAFO QUADRADO

O quadrado de G é o grafo com os mesmos vértices e com as mesmas arestas em adição a (x,y) onde, em G, $d(x, y) = 2$. G^2 denota o quadrado de G. É importante notar que um grafo G pode ter várias raízes quadradas. A figura 2.14 descreve um grafo e seu grafo quadrado. O grafo quadrado possui as arestas $(0, 2)$ e $(1, 3)$ a mais que o grafo.

Figura 2.14: exemplo de grafo e seu grafo quadrado.

3 PROPOSTA

Este trabalho tem como proposta estudar o problema de encontrar os grafos G, tal que, dado um grafo H, $H \approx (L(G))^2$.

Buscar os grafos G dentro de todos os possíveis grafos seria muito custoso. Portanto, buscaremos, a priori, as raízes quadradas de H. Chamaremos as raízes quadradas de $H,_1, H,_2, \ldots, H,_k$, sendo $H,_1$ a primeira raiz quadrada de H e $H,_k$ a última raiz quadrada de H.

Para cada $H,_i$, com $1 \leq i \leq k$, encontramos um grafo I_i correspondente que é a pré-imagem única bipartida do grafo linha de $H,_i$, ou seja, $H,_i \approx L(I_i)$, caso exista.

Então, verificamos se I_i é um grafo bipartido de permutação. Se I_i é um grafo bipartido de permutação, então I_i é uma das possíveis respostas para o problema.

4 RAIZ QUADRADA DE GRAFO

Um grafo qualquer pode ter várias raízes, conforme o descrito na definição de grafo quadrado. Neste trabalho, listamos as raízes. Encontrar as raízes quadradas de um grafo com cintura menor ou igual 5 é um problema NP-completo, tal qual provado em (Farzad et al., 2012b) e (Farzad et al., 2012a). As raízes que procuramos são grafos linha de grafos bipartidos de permutação.

Seja L um grafo, tal que $G = L^2$ e L seja o grafo linha de um grafo bipartido de permutação. Em um grafo linha, um vértice de grau k gera uma clique de ordem k. Ciclos se mantêm no grafo linha. Portanto, se existir um ciclo de tamanho menor ou igual a 4, temos um ciclo de mesmo tamanho no grafo linha. E, portanto, a cintura é menor ou igual a 4. Se existir um vértice de grau maior ou igual a 3, o grafo linha tem uma clique de tamanho maior ou igual a 3 e, portanto, tem triângulos. Logo, a cintura é 3. Para a cintura de L ser maior que 4, um grafo H, tal que H é bipartido de permutação e $L = L(H)$, deve ser acíclico. Para L ter cintura maior que 3, H tem grau máximo 2. Assumindo que H é conexo, então deve ser um caminho ou um ciclo. E esse ciclo deve ser de tamanho 4. Então o grafo L pode ter:

a cintura de tamanho 3, onde H tem algum vértice com grau maior que 2;

b cintura de tamanho 4, onde tanto H quanto L tem um ciclo de tamanho 4;

c cintura de tamanho infinito, onde tanto H quanto L é um caminho.

4.1 ALGORITMO DE RAIZ QUADRADA DE GRAFO

Uma ideia simples de algoritmo para encontrar raízes quadradas de um grafo G qualquer é remover arestas do grafo G, elevá-lo ao quadrado e verificar se o resultado é igual a G. $GERA\_GRAFO(x, y)$ é uma função que gera um grafo a partir do conjunto de vértices x e do conjunto de arestas y.

Algoritmo 1: Encontrar raízes quadradas

```
raizes ← {}
RAIZ_QUADRADA_GRAFO(G, V(G), E(G))
Function RAIZ_QUADRADA_GRAFO()
    G2 ← GRAFO_QUADRADO(GERA_GRAFO(V,E))
    se G2 = G então
        adicione GERA_GRAFO(V, E) a raizes
    se G for subgrafo de G2 então
        para cada e ∈ E faça
            RAIZ_QUADRADA_GRAFO(G, V, E − e)
```

4.2 EXEMPLO DE EXECUÇÃO DO ALGORITMO DE RAIZ QUADRADA DE GRAFO

O grafo G de entrada possui $V(G) = \{1, 2, 3, 4, 5\}$ e $E(G) = \{(1, 2), (1, 3), (1, 4), (1, 5), (2, 3), (3, 4), (3, 5), (4, 5)\}$. Estes dois conjuntos são passados como parâmetros do algoritmo. A subfigura I da figura 4.1 descreve G. Cada parágrafo a seguir descreve, respectivamente, o que ocorre em cada uma das subfiguras, começando a partir da subfigura II.

Executando o algoritmo em G, removemos primeiro a aresta $(1, 2)$ do conjunto E. Em $G2$, pelo menos a aresta $(2, 4)$ aparece. Como essa aresta não está em $E(G)$, o grafo $G(V, E)$ não pode ser uma raiz.

Continuamos a execução e removemos a aresta , $(1, 3)$ de E. Em $G2$, a aresta $(1, 2)$ não tem como aparecer, pois 1 e 2 têm distância maior que 2. Para G ser subgrafo de $G2$, todas as arestas de G devem estar em $G2$. Como $E(G)$ tem uma aresta fora de $E(G2)$, vamos continuar a execução a partir de outro grafo.

Continuamos a execução a partir do grafo cujo conjunto $E = \{ (1, 3) , (1, 4) , (1, 5) , (2, 3) , (3, 4) , (3, 5) , (4, 5) \}$. Removemos a aresta $(1, 4)$. $E(G2)$ tem a aresta $(2, 4)$. Portanto, $G(V, E)$ não é raiz.

Continuamos a execução removendo a aresta $(1, 5)$. $E(G2)$ tem a aresta $(2, 4)$. Portanto, $G(V, E)$ não é raiz.

Continuamos a execução removendo a aresta $(3, 4)$. $E(G2)$ não tem a aresta $(1, 4)$. Portanto, $G(V, E)$ não é raiz e G não é subgrafo de $G2$.

Continuamos a execução a partir do grafo cujo conjunto $E = \{ (1, 3) , (1, 4) , (1, 5) , (2, 3) , (3, 4) , (3, 5) , (4, 5) \}$. Removemos a aresta $(3, 5)$. $E(G2)$ tem a aresta $(2, 4)$. Portanto, $G(V, E)$ não é raiz.

Continuamos a execução removendo a aresta $(3, 4)$. $E(G2)$ tem as mesmas arestas de $E(G)$. Portanto, $G(V, E)$ é raiz de G.

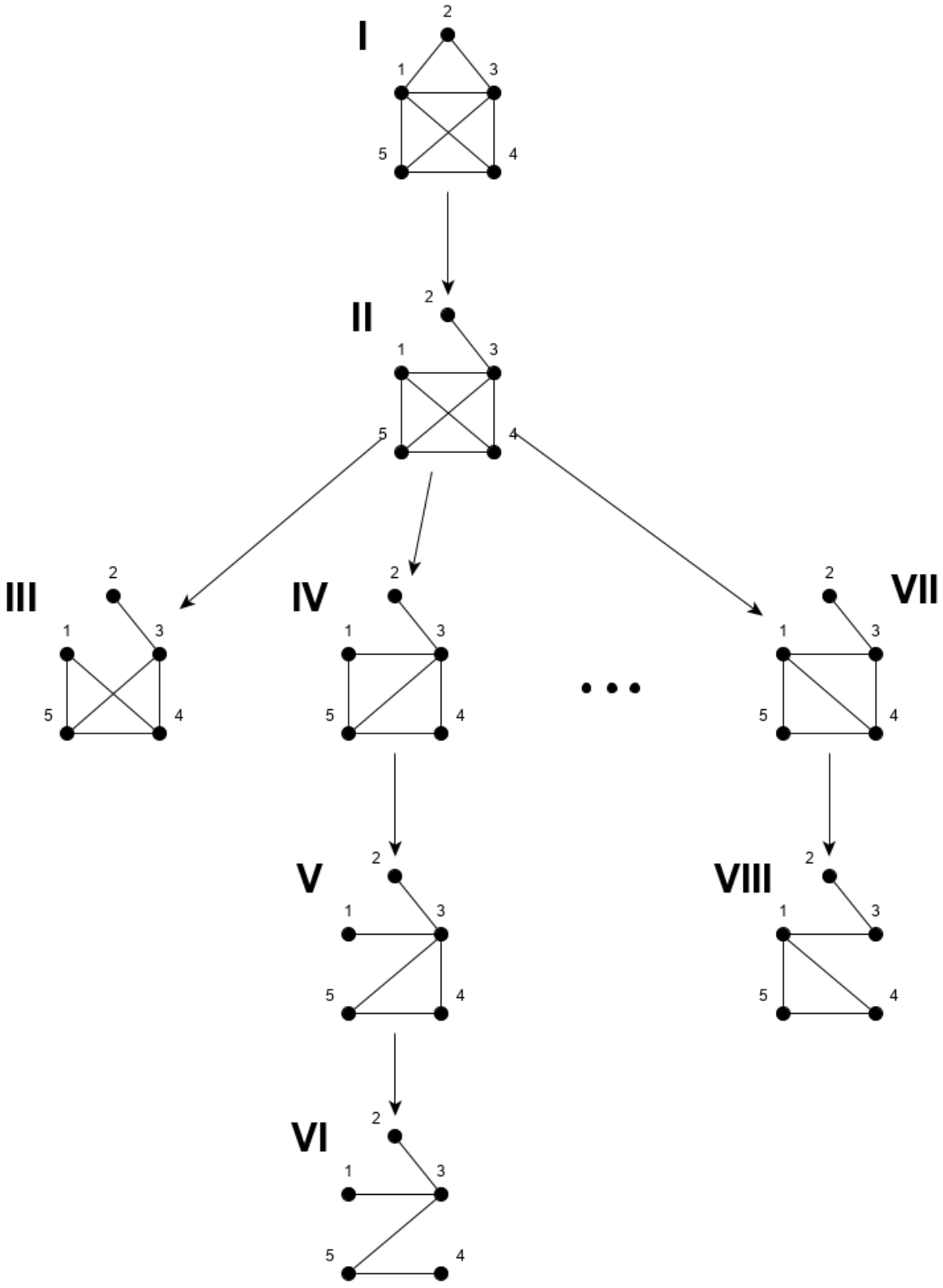


Figura 4.1: árvore de execução do algoritmo de raiz quadrada.

5 PRÉ-IMAGEM DE GRAFO LINHA

Encontrar a pré-imagem de um grafo linha é um problema polinomial, resolvido em tempo linear. A pré-imagem de um grafo linha é única quando se considera isomorfismo de grafos, com exceção de dois casos. Quando a pré-imagem possui um triângulo, um subgrafo completo com 3 vértices, ou um vértice de grau 3, o grafo linha tem um triângulo.

O algoritmo de (Lehot, 1974) resolve esse problema de ambiguidade verificando se há vértices cruzados no grafo. Vértices cruzados indicam que há um triângulo na pré-imagem.

5.1 ALGORITMO DE PRÉ-IMAGEM DE GRAFO LINHA

O algoritmo a seguir para encontrar a pré-imagem de grafo linha é uma transliteração do presente no artigo de (Lehot, 1974).

É necessário definir o que são triângulos pares e triângulos ímpares para compreender o algoritmo. Um triângulo par é que dado um vértice $v \in V$, v é adjacente a zero ou dois vértices do triângulo. Um triângulo ímpar é um triângulo que não é par. O cume de um triângulo é o vértice de um triângulo ímpar que ainda não foi nomeado.

Uma aresta é nomeada como (x, y) se ligar o vértice x ao vértice y. Como um grafo linha é grafo de intersecção de arestas, o vértice que representa a aresta (x, y) é nomeado como $x - y$. Durante o processo de nomeação, podemos não saber o nome de um das pontas das arestas em determinado ponto da execução. Portanto, nomeamos o vértice parcialmente como $x-$ ou $y-$. Um vértice não nomeado completamente é um vértice sem nome ou nomeado parcialmente.

A $LISTA$ no passo nove é uma lista com todos os vértices não nomeados completamente.

Em (Lehot, 1974), o algoritmo leva em conta um grafo, H, isomorfo a pré-imagem linha, G no contexto do algoritmo, que procuramos para verificar se o grafo é linha. Diferente do algortimo, vamos encontrar a pré-imagem a partir do grafo linha completamente nomeado e passar para o passo seguinte.

Algoritmo de (Lehot, 1974)

Passo 1. Escolhemos dois vértices adjacentes. Nomeamo-los de $1 - 2$ e $2 - 3$. Eles são os dois vértices básicos escolhidos inicialmente e definem a clique 2, ou seja, as arestas que incidem sobre o vértice 2 no grafo da pré-imagem.

Passo 2. Encontramos todos os vértices adjacentes a ambos os vértices básicos. Se houver três ou mais vértices, vamos imediatamente para o passo 5. Se não houver, vamos para o passo 6.

Um problema importante é determinar qual é, se houver, o vértice cruzado e, dependendo da quantidade de vértices adjacentes a ambos os vértices, consideramos um de três casos: quando há apenas um vértice, passo 3; quando há dois vértices, passo 4; e quando há três ou mais vértices, passo 5.

Passo 3. Há apenas um vértice. Chamamo-lo de x. Se existir mais um vértice adjacente a outro vértice do triângulo sem ser adjacente a qualquer outro vértice do triângulo, então $x = 2 - 4$. Em seguida, vamos para o passo 6. Caso contrário, $x = 1 - 3$ e vamos para o passo 7.

Nesse passo, são considerados dois subcasos: se o triângulo formado por $(x, 1 - 2, 2 - 3)$ é ímpar ou é par. Se for ímpar, ou seja, se houver um vértice que é adjacente a um ou três vértices do triângulo, x deve ser $2 - 4$. Se x fosse $1 - 3$, ele obrigatoriamente seria adjacente a dois vértices e o triângulo é par.

Passo 4. Há dois vértices adjacentes a ambos os vértices básicos do primeiro passo. Chamamo-los de x e y. Se x e y são adjacentes, então não há vértice cruzado. Vamos para o passo 6. Se eles não são adjacentes, eles constituem dois triângulos com os vértices básicos, e verificamos se há um triângulo ímpar. Se houver um, então o cume deve ser $2-4$. Se x é o cume, então $x = 2-4$ e $y = 1-3$ e y é o vértice cruzado. Vamos para o passo 6. Caso contrário, ambos os triângulos são pares. A figura 5.1 mostra as três possibilidades restantes.

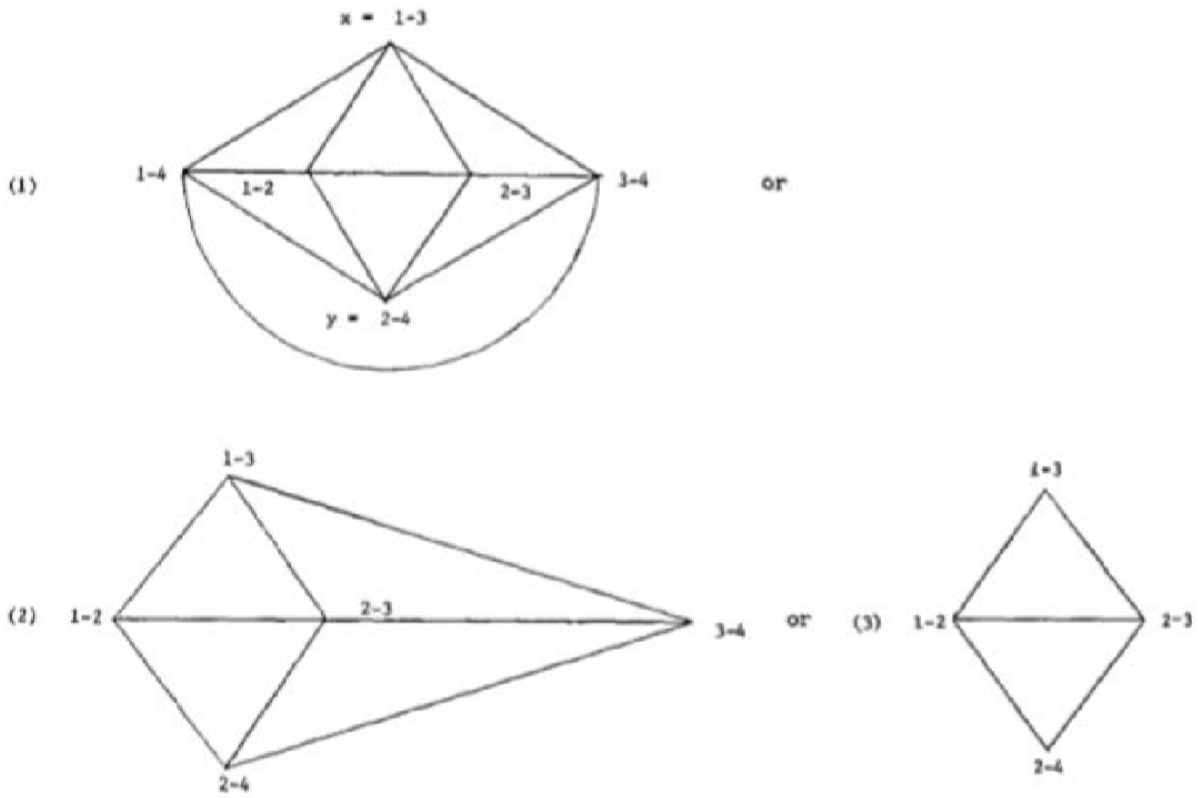


Figura 5.1: 3 possibilidades restantes do passo 4 do algoritmo do Lehot. p. 574

Se x e y são adjacentes, eles formam uma clique com $1-2$ e $2-3$, o que significa que não existe triângulo no grafo da pré-imagem e, portanto, não existe vértice cruzado. Se não forem, são formados dois triângulos: $(x, 1-2, 2-3)$ e $(y, 1-2, 2-3)$. Se um deles é ímpar, então as regras do passo anterior se aplicam, e o vértice não nomeado é $2-4$. Consequentemente, o outro será $1-3$. Caso contrário, ambos são pares, e a figura 5.1 cobre os casos restantes.

Passo 5. Há um grupo de três ou mais vértices adjacentes aos vértices básicos. Se houver um vértice α do grupo que não é adjacente a um vértice β(escolhido aleatoriamente, mas uma única vez), então ou α é o primeiro vértice do grupo sob investigação, ou não é. No primeiro caso, o impasse é quebrado examinando a adjacência de α a um terceiro vértice do grupo. Se for adjacente, então β é declarado o vértice cruzado. Se não for, então α é declarado o vértice cruzado. Se não houver vértice cruzado, vamos para o passo 6. Se houver um, vamos para o passo 7.

Passo 6. Todos os vértices da clique foram nomeados e eles, sucessivamente, nomeiam parcialmente todos os vértices "ainda-não-completamente-nomeados" que são adjacentes a eles. Vamos para o passo 8.

Passo 7. Todos os vértices da clique foram nomeados e o vértice cruzado está nomeado completamente. Apenas os vértices da clique serão usados para, sucessivamente, nomear

parcialmente os vértices “ainda-não-completamente-nomeados” que são adjacentes. Então, as duas cliques associadas são desprezadas e o processo de nomeação continua normalmente. Vamos para o passo 8.

Passo 8. Escolhemos um novo par de vértices básicos. Se todos os vértices estiverem completamente nomeados, vamos para o passo 9. Escolhemos qualquer vértice nomeado parcialmente. Ele é adjacente a um vértice completamente nomeado. Se esse vértice completamente nomeado não pertence a nenhuma clique já descoberta, então há um vértice completamente nomeado que não pertence a uma clique já descoberta(que nomeou parcialmente o vértice parcialmente nomeado) e que é definido como o vértice cruzado entre os dois vértices anteriores. Portanto, o vértice cruzado é encontrado em um ou dois passos. Além disso, o último par de vértices envolvidos no processo de nomeação parcial poderia ser escolhido do topo de uma pilha para esse propósito. O par de vértices básicos nomeia completamente o vértice básico que estava apenas nomeado parcialmente. Descobrimos esses vértices, dentro da LISTA de vértices ainda não completamente nomeados, que são adjacentes a ambos os vértices básicos e nomeamo-los parcialmente com o número da clique em questão, e completamos seus nomes. Todos os vértices da clique agora estão completamente nomeados. Removemo-los da LISTA e nomeamos parcialmente todos os vértices adjacentes a vértices da clique com o número, que não é o número da clique, em seus nomes. Então, vamos para o passo 8 de novo.

Passo 9. Marcamos as arestas de $L(G)$, o grafo linha de G, que estão em H, até que uma aresta de H não esteja em $L(G)$. Se isso ocorrer, então H não é um grafo linha. Caso contrário, a saída é que H é um grafo linha.

5.2 EXEMPLO DE EXECUÇÃO DO ALGORITMO DE PRÉ-IMAGEM DE GRAFO LINHA

Dado um grafo G, mostrado na figura 5.2, começamos escolhendo dois vértices para iniciar o processo de nomeação.

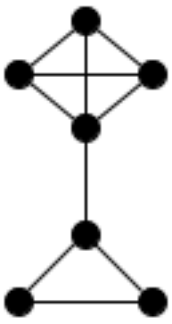


Figura 5.2: grafo linha sem nenhum vértice nomeado.

Executando o passo 1 do algoritmo, podemos observar que os vértices que foram escolhidos para ser os vértices básicos $1-2$ e $2-3$ na figura 5.3, que denotam a clique 2. Vamos para o passo 2.

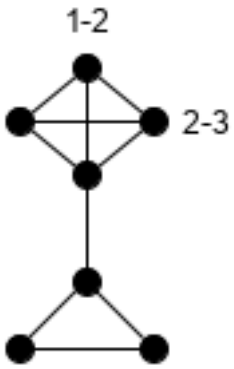


Figura 5.3: vértices básicos 1-2 e 2-3 foram escolhidos.

Após nomeá-los, verificamos o número de vértices adjacentes a ambos os vértices básicos, executando o passo 2 do algoritmo. No grafo G, dado os vértices que escolhemos, existem dois. Portanto, o passo 4 é executado. Nomeamos os dois como x e y.

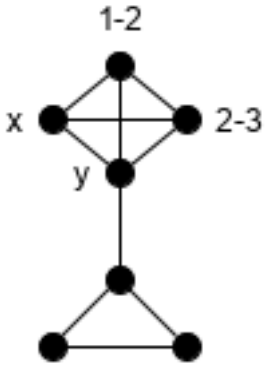


Figura 5.4: foram identificados dois vértices adjacentes aos vértices básicos.

Como formam uma clique com $1-2$ e $2-3$, os vértices x e y também pertencem à clique 2. Logo, podemos nomear x como $2-4$ e y como $2-5$. Vamos para o passo 6.

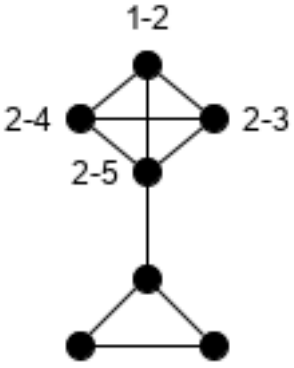


Figura 5.5: clique 2 completamente nomeada.

No passo 6, nomeamos todos os vértices que são adjacentes à clique. Um único vértice, nomeado parcialmente como 5−, é adjacente à clique. Vamos para o passo 8.

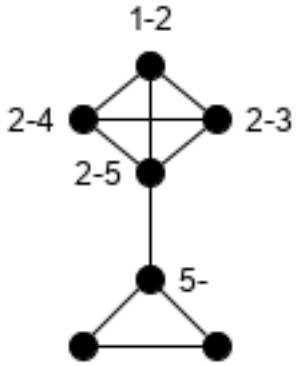


Figura 5.6: vértice adjacente à clique 2 parcialmente nomeado como 5-.

Nomeamos o vértice parcialmente nomeado da clique 5 como $5-6$. Após a clique 5 estar completamente nomeada, nomeamos parcialmente os vértices adjacentes a ela. Eles são nomeados como 6−, pois a clique 5 é composta por $2-5$ e $5-6$ e a clique da vez é a clique 6.

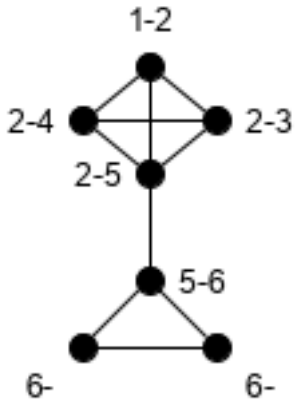


Figura 5.7: clique 6 parcialmente nomeada.

Ainda no passo 8, devemos terminar de nomear a clique 6. Nomeamos os dois vértices restantes como $6-7$ e $6-8$.

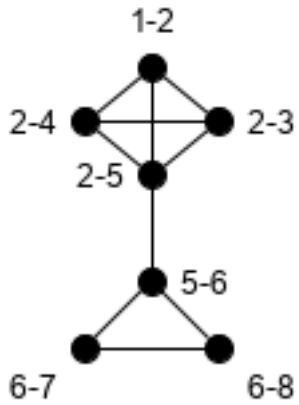


Figura 5.8: grafo linha completamente nomeado.

A pré-imagem adquirida a partir do grafo *G* dado é mostrada na figura 5.9. Os vértices do grafo linha indicam quais são as arestas presentes na pré-imagem. Sabendo que há 8 vértices na pré-imagem, pois o algoritmo terminou com 8 sendo o número mais alto do processo de nomeação, basta ligá-los de acordo com os nomes dos vértices do grafo linha.

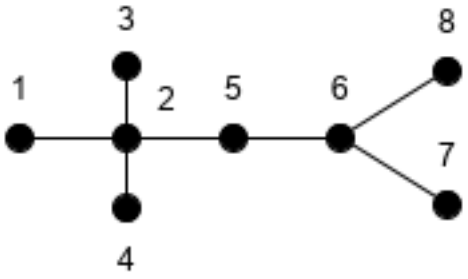


Figura 5.9: pré-imagem do grafo linha.

6 GRAFO BIPARTIDO DE PERMUTAÇÃO

6.1 ALGORITMO DE RECONHECIMENTO DE GRAFO BIPARTIDO DE PERMUTAÇÃO

O algoritmo de reconhecimento de grafo bipartido de permutação está descrito com mais detalhes em (Spinrad et al., 1987). O tempo de execução deste algoritmo é da ordem $O(|V| + |E|)$. Este capítulo é apenas um resumo para compreender como funciona o algoritmo de reconhecimento.

A ideia principal do algoritmo é, dado um grafo bipartido $G(\mathrm{A}, \mathrm{B}, E)$, manter uma estrutura de dados que possua tanto a *propriedade da adjacência* quanto a *propriedade do enclausuramento*. Os vértices de B são examinados um de cada vez e cada um deles força mais restrições para a permutação.

São necessárias algumas novas terminologias para explicar o algoritmo.

Um bloco S é um conjunto de vértices de A que deve ocorrer consecutivamente em quaisquer ordenações de A e que possui tanto a *propriedade da adjacência* quanto a *propriedade do enclausuramento*, mas vértices em S podem ocorrer em qualquer ordem.

Uma lista de blocos é uma sequência de blocos $S_1, S_2, ..., S_k$. Vértices dentro de cada bloco ocorrem consecutivamente na nossa permutação de A, e o último vértice de S_i precederá o primeiro vértice de $S_{(i+1)}$, $\forall \leq i \leq k - 1$.

Conforme cada vértice de B é examinado, há uma lista de blocos vigente, que representa um conjunto mínimo de restrições em qualquer ordenação de A que possui tanto a *propriedade da adjacência* quanto a *propriedade do enclausuramento*. Essa lista de blocos vigente é chamada de $SLIST$. Vértices de A são chamados de novos se eles não estão em nenhum bloco de $SLIST$, e são velhos se eles já estão em algum bloco de $SLIST$. O bloco mais a esquerda de $SLIST$ será chamado S_L, e o bloco mais a direita de $SLIST$ será chamado de S_R.

Para cada vértice β em B, seja $ADJ(\beta)$ o conjunto de blocos em $SLIST$ que contêm pelo menos um vértice de $N(\beta)$. Seja $MIX(\beta)$ o conjunto de blocos que contêm pelo menos um vértice de $N(\beta)$ e pelo menos um vértice de $\mathrm{A} - N(\beta)$.

No começo do algoritmo, encontramos o vértice y em B tal que $|N(y)|$ é mínimo. Criamos um bloco contendo os membros de $N(y)$, e deixamos nossa $SLIST$ inicial ser esse bloco.

Para cada passo restante do algoritmo, consideramos algum vértice β de B que é adjacente a pelo menos um vértice velho. Modificamos $SLIST$ para que os vértices adjacentes a β ocorram consecutivamente e não 'enclausurem' os vizinhos de qualquer outro vértice ou sejam 'enclausurados' pelos vizinhos de qualquer outro vértice, se isso for possível.

Nós dividimos a descrição do algoritmo em quatro casos, dependendo da composição de $N(\beta)$.

Caso 1: $N(\beta)$ contém um novo vértice.

- Caso 1.1: Algum vértice velho não está em $N(\beta)$.
- Caso 1.2: $N(\beta)$ contém todos os vértices velhos.

Caso 2: Todos os vértices em $N(\beta)$ são velhos.

- Caso 2.1: $N(\beta)$ contém vértices de dois blocos distintos de $SLIST$.
- Caso 2.2: $N(\beta)$ está completamente contida em um único bloco de $SLIST$.

6.1.1 Caso 1: $N(\beta)$ contém um vértice novo.

Para ambos os casos 1.1 e 1.2, criamos um bloco S_{novo} constituído de todos os vértices novos em $N(\beta)$.

6.1.1.1 Caso 1.1*: Algum vértice velho não está em* $N(\beta)$*.*

Primeiramente, decidimos onde os novos vértices pertencem. E, então, modificamos os vértices velhos para refletir as restrições impostas por β.

Se $SLIST$ tem apenas um bloco, arbitrariamente colocamos S_{novo} no fim de $SLIST$. Se $SLIST$ tiver mais de um bloco, é fácil determinar qual lado de $SLIST$ o bloco S_{novo} deve ser adicionado. Os blocos de $SLIST$ devem ocorrer consecutivamente, assim como os vértices de $N(\beta)$. Logo, S_{novo} pertence ou do lado esquerdo ou do lado direito de $SLIST$. Se S_L não está em $ADJ(\beta)$, ou se S_L está em $MIX(\beta)$ e S_R está em $ADJ(\beta)$, então S_{novo} é colocado no fim de $SLIST$. Caso contrário, S_{novo} é colocado no começo de $SLIST$. Esse posicionamento é forçado pela *propriedade da adjacência* a vizinhos de β.

Depois de S_{novo} ser posicionado, devemos modificar os blocos velhos de $SLIST$ para que os vértices de $N(\beta)$ sejam forçados a ocorrer consecutivamente. Se houver qualquer bloco S em $MIX(\beta)$, removemos os vértices de S que estão em $N(\beta)$ de S, criando um novo bloco R. S não pode estar entre R e S_{novo} se a *propriedade da adjacência* estiver satisfeita, então R é colocado no lado de S que for mais próximo de S_{novo}.

6.1.1.2 Caso 1.2*: $N(\beta)$ contém todos os vértices velhos.*

Nesse caso, o algoritmo simplesmente coloca S_{novo} no fim de $SLIST$.

Se houver apenas um bloco em $SLIST$ nesse momento, essa escolha é completamente arbitrária. Se houver mais de um bloco em $SLIST$, a escolha é forçada para que $N(\beta)$ não enclausure $N(y)$, onde y foi o primeiro vértice usado para criar o bloco inicial. O algoritmo sempre coloca pelo menos um vértice s que não está em $N(y)$ a direita de $SLIST$ quando o segundo bloco de $SLIST$ é criado. Já que $N(\beta) - N(y)$ inclui tanto s quanto S_{novo}, S_{novo} também deve ser colocado a direita de $SLIST$.

6.1.2 Caso 2:Todos os vértices em $N(\beta)$ são velhos.

6.1.2.1 Caso 2.1*: $N(\beta)$ contém vértices de dois blocos distintos de $SLIST$.*

Precisamos garantir que os vértices de $N(\beta)$ ocorram consecutivamente. Para qualquer bloco S que esteja em $MIX(\beta)$, removemos os vértices de $N(\beta)$ de S para formar um novo bloco R. Inserimos R entre S e o bloco mais próximo de S em $SLIST$ que também está em $ADJ(\beta)$. Se existem dois blocos vizinhos a S em $ADJ(\beta)$, é fácil perceber que não há arranjo possível que tenha a *propriedade da adjacência*, então essa é a única maneira possível de refinar a lista de blocos.

6.1.2.2 Caso 2.2*: $N(\beta)$ está completamente contida em um único bloco S de $SLIST$.*

Se todos os vértices de S estão em $N(\beta)$, nada ocorre. Caso contrário, removemos os vértices de $N(\beta)$ de S, formando um novo bloco R. Se $S = S_R$, então colocamos R no fim de $SLIST$. Se $S = S_L$, colocamos R no início de $SLIST$. Se S não é S_L ou S_R, então o grafo não é um grafo bipartido de permutação.

6.2 EXEMPLO DE EXECUÇÃO DO ALGORITMO DE RECONHECIMENTO DE GRAFOS BIPARTIDOS DE PERMUTAÇÃO

O grafo G da entrada é o que está presente na figura 6.1. O grafo G tem conjunto A = $\{1, 2\}$, conjunto B = $\{3, 4, 5, 6\}$ e conjunto $E(G) = \{(1, 3), (1, 4), (2, 3), (2, 4), (2, 5), (2, 6)\}$. O conjunto VV denota os vértices visitados durante o algoritmo. No começo do algoritmo,

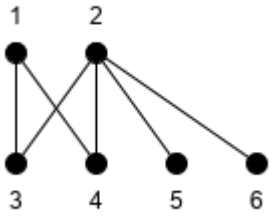


Figura 6.1: grafo de entrada para algoritmo de reconhecimento de grafos bipartidos de permutação.

devemos escolher um vértice y de B tal que $|N(y)|$ é mínimo. Tanto 5 quanto 6 suprem essa condição, pois tem apenas um vizinho. Caso haja várias escolhas possíveis, faremos escolhas por ordem crescente. Como $5 < 6$, escolhemos o vértice 5. Como visitamos o vértice 5, $VV = \{5\}$. Portanto, $SLIST$ é composta por um bloco S_1 que contém o vértice 2. $SLIST = \{\{2\}\}$.

Devemos escolher um vértice que seja vizinho de um de vértice velho. Há três disponíveis. Mantendo as escolhas por ordem crescente, o vértice 3 é o escolhido. $VV = \{5, 3\}$. Na vizinhança de 3 estão os vértices 1 e 2. Esse é o caso 1.2, onde $N(\beta)$ contém um vértice novo e contém todos os vértices velhos. Nesse caso, adicionamos um bloco S_2, que contém o vértice 1, no fim de $SLIST$. $SLIST = \{\{2\}, \{1\}\}$.

O próximo vértice de B a ser visitado é 4. Ele cai no caso 2.1. Devemos remover $N(\beta)$ de $MIX(\beta)$, criar um bloco R e adicioná-lo entre S e o bloco seguinte a S. Como $MIX(4)$ é nulo, não fazemos nada nesse passo e seguimos a execução. $VV = \{5, 3, 4\}$. O vértice restante é o vértice 6 e ele cai no caso 2.2, no qual não se faz nada. $VV = \{5, 3, 4, 6\}$. Com $SLIST = \{\{2\}, \{1\}\}$, a *propriedade da adjacência* e a *propriedade do enclausuramento* são satisfeitas. Portanto, o grafo é um grafo bipartido de permutação.

7 CONCLUSÃO

Encontrar a pré-imagem de um grafo biclique é um problema NP-completo. Encontrar a raiz quadrada de um grafo também. Todavia, para alguns casos, como os da figura 7.1, encontrar uma raiz que satisfaça o resto do algoritmo é simples.

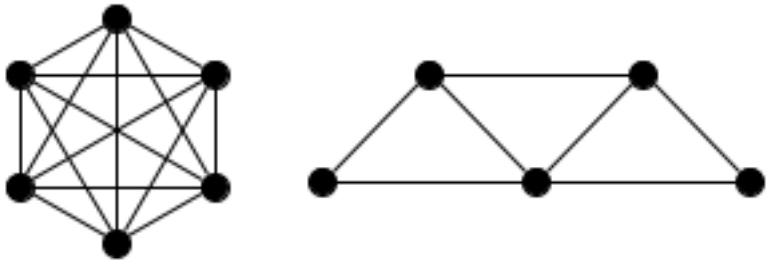


Figura 7.1: grafo completo e grafo com 3 triângulos encadeados.

No caso do grafo completo, uma raiz possível é o próprio grafo completo. Executar o resto do algoritmo nele também é trivial, pois encontrar a pré-imagem de grafo linha de um grafo completo também é simples. É um grafo estrela com o número de vértices do grafo completo, pois todos pertencem a mesma clique. E todo grafo estrela é um grafo bipartido de permutação. O segundo grafo da imagem 7.1 tem apenas uma raiz, que é um caminho entre cada um dos vértices. A pré-imagem de grafo linha de um caminho é o mesmo grafo com um vértice, de uma das pontas, a menos. E todo caminho é um grafo bipartido de permutação.

Trabalhos futuros podem incluir possíveis melhorias ao algoritmo. Se houver um vértice cruzado durante a execução do algoritmo de pré-imagem de grafo linha, podemos parar a execução e continuar a buscar outra raiz no passo anterior. Ter um vértice cruzado implica em ter um triângulo na pré-imagem. Nenhum grafo que possui um triângulo, pois um triângulo é um ciclo ímpar, é bipartido. Logo, não pode ser bipartido de permutação. É possível, também, implementar no algoritmo de raiz quadrada uma forma de identificar casos fáceis de achar a raiz, como grafos completos e grafos que são triângulos encadeados, para torná-lo mais eficiente nesses casos.

REFERÊNCIAS

Cruz, E. P., Groshaus, M., Guedes, A. e Puppo, J. (2018). Biclique graph of interval bigraphs. Submitted.

Farzad, B., Lau, L. C., Le, V. B. e Tuy, N. N. (2012a). Complexity of finding graph roots with girth conditions. *Algorithmica*, 62(1-2):38–53.

Farzad, B., Lau, L. C., Le, V. B. e Tuy, N. N. (2012b). Complexity of finding graph roots with girth conditions. *Algorithmica*, 62(1-2):38–53.

Groshaus, M. e Guedes, A. L. P. (2018). The biclique graph of K_3-free graphs are the square of some graph. Em *Proc. VIII Latin American Workshop on Cliques in Graphs*, página 25, Rio de Janeiro.

Lehot, P. G. H. (1974). An Optimal Algorithm to Detect a Line Graph and Output Its Root Graph. *J. ACM*, 21(4):569–575.

Spinrad, J., Brandstädt, A. e Stewart, L. (1987). Bipartite permutation graphs. *Discrete Applied Mathematics*, 18(3):279–292.