CRISTOPHER CARCERERI

RAIZ QUADRADA COM CINTURA PELO MENOS SEIS DE UM GRAFO

(*versão pré-defesa, compilada em 19 de janeiro de 2024*)

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: Prof. Dr. Renato Carmo.

Coorientador: Prof. Me. Aleffer Rocha.

CURITIBA PR

2023

RESUMO

O quadrado de um grafo H é o grafo H^2 obtido adicionando-se a H arestas entre todos os vértices a distância 2. Se $H^2 = G$, dizemos que H é uma raiz quadrada de G. Determinar se um grafo tem raiz quadrada é um problema $\mathcal{NP}$-completo. No entanto, são conhecidos algoritmos de tempo polinomial para encontrar raízes pertencentes a certas classes. Aqui, consideramos os problemas de encontrar uma raiz quadrada com cintura pelo menos k de um grafo G para $k \in \{6, 7\}$. As melhores soluções publicadas são um algoritmo $O(\delta(G) \cdot n^4)$ para $k = 6$ e um algoritmo $O(m \cdot n^2)$ para $k = 7$ (sendo m e n o número de arestas e vértices de G, respectivamente). Mostramos que é possível resolver esses problemas em tempo $O(\delta(G) \cdot n^2)$ para $k = 6$ e $O(n^2)$ para $k = 7$.

Palavras-chave: Raiz quadrada de grafo. Complexidade de algoritmos. Detecção de ciclo.

ABSTRACT

The square of a graph H is the graph H^2 obtained by adding to H edges joining all vertices at distance 2. If $H^2 = G$, we say that H is a square root of G. To decide if a graph has a square root is an $\mathcal{NP}$-complete problem. However, polynomial-time algorithms for finding roots belonging to certain classes are known. Here we consider the problems of finding a square root of girth at least k of a graph G, for $k \in \{6, 7\}$. The best published solutions are an $O(\delta(G) \cdot n^4)$ algorithm for $k = 6$ and an $O(m \cdot n^2)$ algorithm for $k = 7$ (m and n being the number of edges and vertices of G, respectively). We show that it is possible to solve these problems in time $O(\delta(G) \cdot n^2)$ for $k = 6$ and $O(n^2)$ for $k = 7$.

Keywords: Graph square root. Algorithm complexity. Cycle detection.

SUMÁRIO

1 INTRODUÇÃO

Um grafo G é dito a *k-ésima potência* de um grafo H — e H uma *raiz k-ésima* de G — se e somente se:

- $V(G) = V(H)$
- $E(G) = \{uv : 1 \leq d_H(u, v) \leq k\}$

Nesse caso, G é denotado H^k. Como usual, com $k = 2$ dizemos que H é uma *raiz quadrada* de G e que G é o *quadrado* de H. Note que não se garante nem a existência nem a unicidade das raízes de um grafo. Por exemplo, na Figura 1.1, os grafos H_1, H_2, H_3 e o próprio grafo G são todos raízes quadradas de G. H_1, H_2 e H_3, por outro lado, não têm raiz alguma.

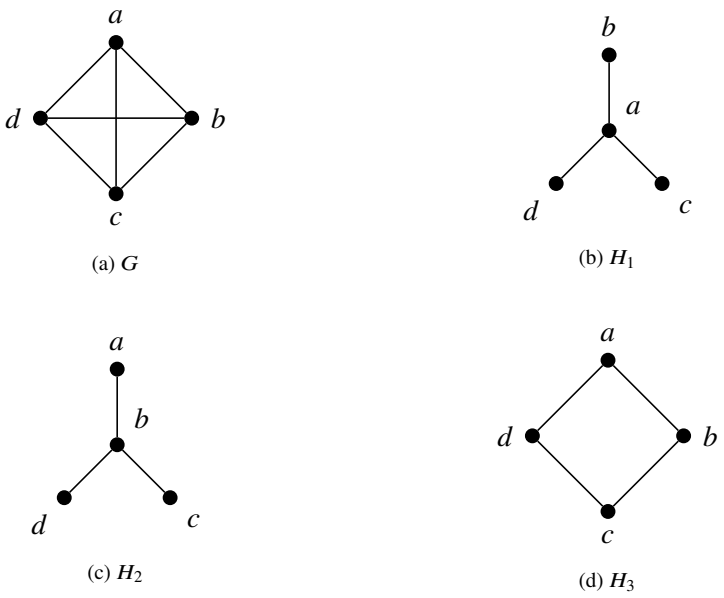

Figura 1.1: Algumas raízes quadradas de K_4

Há alguns resultados interessantes sobre potências e raízes. Por exemplo, Sejanina (1960) mostra que para todo grafo G não trivial e conexo, G^3 é hamiltoniano. Fleischner (1974) prova que o quadrado de um grafo 2-conexo também tem ciclo hamiltoniano. O algoritmo em Lau (1980) encontra esse ciclo em tempo polinomial. Foram propostas aproximações para o Problema do Caixeiro Viajante com desigualdade triangular parametrizada[1] que usam esses ciclos hamiltonianos (Bender e Chekuri, 2000; Andreae e Bandelt, 1995; Andreae, 1995). Também há aplicações na biologia computacional para reconstrução de árvores filogenéticas (Lin et al., 2000; Nishimura et al., 2002).

[1]Uma desiguldade triangular relaxada, i.e., uma desiguldade da forma $a \leq \tau(b + c)$.

A k-ésima potência de um grafo pode ser encontrada por buscas de profundidade limitada em k ou multiplicações de matrizes. Na outra direção, considere o *Problema da Raiz Quadrada*:

Problema da Raiz Quadrada

Instância: *Um grafo G.*
Pergunta: *Existe um grafo que seja raiz quadrada de G?*

Seja G um grafo com n vértices e m arestas. Ross e Harary (1960) mostra que se existe uma árvore que seja raiz quadrada de G, ela é única a menos de isomorfismo. Lin e Skiena (1991) apresenta um algoritmo para encontrar uma dessas árvores em tempo $O(m+n)$, mas conjectura que o Problema da Raiz Quadrada é $\mathcal{NP}$-completo, o que foi provado por Motwani e Sudan (1994). A redução usada na prova de Motwani e Sudan (1994) implica também que é $\mathcal{NP}$-completo encontrar uma raiz quadrada com cintura 3.

Farzad et al. (2009) apresenta um algoritmo $O(\delta(G) \cdot n^4)$ para encontrar uma raiz quadrada que tenha cintura 6 ou maior de G e um algoritmo $O(m \cdot n^2)$ para raiz com cintura 7 ou maior, demonstrando, para o segundo caso, sua unicidade a menos de isomorfismo. Os autores demonstraram ainda que decidir se há raiz com cintura 4 é um problema $\mathcal{NP}$-completo. Adamaszek e Adamaszek (2011) mostra a unicidade a menos de isomorfismo da raiz com cintura pelo menos 6 — a melhor extensão possível dos resultados anteriores para árvores e grafos com cintura pelo menos 7.[2] Karimi (2013) prova que decidir se há raiz com cintura 5 é também um ploblema $\mathcal{NP}$-completo.

Há, destarte, uma dicotomia para a complexidade de encontrar uma raiz quadrada com uma cintura $k \geq 3$: o problema é polinomial se $k \geq 6$ e $\mathcal{NP}$-completo caso contrário.

Este trabalho apresenta um algoritmo de menor complexidade — $O(\delta(G) \cdot n^2)$ — para o caso polinomial: computar uma raiz quadrada com cintura 6 ou mais. Além disso, mostramos que é possível encontrar uma raiz quadrada com cintura 7 ou mais em tempo $O(n^2)$.

1.1 DEFINIÇÕES E NOTAÇÃO

Dados um conjunto S e um inteiro k, denotamos por $\binom{S}{k}$ o conjunto dos subconjuntos de S com exatamente k elementos.

Um grafo (*simples*) G é um par $(V(G), E(G))$ onde $V(G)$ é um conjunto finito e $E(G) \subseteq \binom{V(G)}{2}$. Um elemento de $V(G)$ é dito um *vértice* de G e um elemento de $E(G)$ é dito uma *aresta* de G. Convencionamos denotar uma aresta $\{u, v\}$ por uv.

Se $uv \in E(G)$, dizemos que u é *vizinho* de v em G. A *vizinhança* de um vértice v em G, denotada $N_G(v)$, é o conjunto de todos os vizinhos de v em G. A *vizinhança fechada* de v em G é o conjunto $N_G[v] = N_G(v) \cup \{v\}$. O *grau* de um vértice v em G, denotado $\delta_G(v)$, é o

[2] O resultado não pode ser extendido para raízes com cintura pelo menos 5. Como o próprio artigo nota, ${K_{1,4}}^2 = {C_5}^2 = K_5$.

número de vizinhos de v em G. O *grau mínimo* de G é o menor grau dentre seus vértices. Um grafo em que todo vértice é vizinho de todos os demais é chamado um *grafo completo*.

Dado um conjunto $S \subseteq V(G)$, o *subgrafo de G induzido por S* é o grafo $G[S]$ tal que $V(G[S]) = S$ e $E(G[S]) = \binom{S}{2} \cap E(G)$.

Um *passeio* (de v_0 a v_n) em um grafo G é uma sequência $(v_0, ..., v_n)$ de vértices de G tal que, para todo $1 \leq i \leq n$, tem-se $v_{i-1}v_i \in E(G)$. O *tamanho* de um passeio é o número de arestas "percorridas" por ele: um passeio $(v_0, ..., v_n)$ tem tamanho n. A *distância* entre dois vértices u e v em G, denotada $d_G(u, v)$, é o tamanho do menor passeio de u a v em G. Um grafo é dito *conexo* se existe passeio de qualquer um de seus vértices a qualquer outro. Um *componente* de G é um subgrafo conexo maximal de G.

Um *ciclo* em um grafo G é um passeio $(v_0, ..., v_n)$ em G com tamanho maior ou igual a 3 e tal que $v_0 = v_n$ e todos os demais vértices são distintos dois a dois. Indicamos um ciclo de n vértices por C_n. Um grafo sem ciclo é chamado de *acíclico*. A *cintura* de G é o tamanho de um ciclo de tamanho mínimo em G. Se G é acíclico, sua cintura é considerada infinita. Uma *árvore* é um grafo conexo e acíclico.

A *matriz de adjacência* de um grafo G é a matriz M indexada por $V(G) \times V(G)$ tal que $M[u, v] = 1$ se $uv \in E(G)$ e $M[u, v] = 0$ se $uv \notin E(G)$.

O quadrado de um grafo H é o grafo H^2 tal que $V(H^2) = V(H)$ e $E(H^2) = \{uv : 1 \leq d_H(u, v) \leq 2\}$. Uma raiz quadrada de um grafo G é um grafo H tal que $H^2 = G$.

Note que há uma relação entre o quadrado de um grafo e o quadrado de sua matriz de adjacência. Se M_H é a matriz de adjacência de H tomada como uma matriz booleana, então a matriz de adjacência de H^2 é $M_H^2 + M_H$.

1.2 ORGANIZAÇÃO

O texto é organizado em 5 capítulos. O Capítulo 2 apresenta o algoritmo de Farzad et al. (2009) para encontrar uma raiz quadrada com cintura pelo menos 6. No Capítulo 3 é exposta a modificação que reduz sua complexidade temporal. O Capítulo 4 mostra um novo algoritmo para calcular uma raiz com cintura pelo menos 7. O Capítulo 5 contém as considerações finais.

2 O ALGORITMO DE FARZAD ET AL.

Considere o seguinte problema de busca.

***Raiz Quadrada com Cintura pelo menos* 6**

Entrada: *Um grafo G.*
Saída: *Um grafo H com cintura pelo menos 6 tal que $H^2 = G$, se existir, ou "Não tem registro", caso contrário.*

Como mencionado, Farzad et al. (2009) apresentaram uma solução polinomial para esse problema. Por simplicidade, serão considerados apenas grafos conexos. Observe que se um grafo não é conexo, basta aplicar o algoritmo separadamente a seus compontes. As duas proposições a seguir lastreiam a solução.

Proposição 2.1 (Farzad et al., 2009)**.** *Se H é um grafo com cintura pelo menos 6, $uv \in E(H)$ e $G = H^2$, então $G[N_G(u) \cap N_G(v)]$ tem no máximo dois componentes: um induzido pelos vértices em $N_H(v) - \{u\}$, outro induzido pelos vértices em $N_H(u) - \{v\}$.* □

Proposição 2.2 (Farzad et al., 2009)**.** *Se um grafo conexo H é livre de $\{C_3, C_5\}$ e $G = H^2$, então para todo $v \in V(H)$ e para todo $u \in N_H(v)$,*

$$N_H(u) = N_G(u) \cap (N_G[v] - N_H(v)).$$ □

A Proposição 2.2 permite, a partir de um grafo G e da vizinhança de um vértice em uma raiz quadrada sem $\{C_3, C_5\}$ de G, determinar também a vizinhança de todos os seus vizinhos. Com base nisso, é possível criar um procedimento similar a uma busca em largura que computa toda a raiz. O Algoritmo 1 recebe como entrada um grafo G, um vértice $v \in V(G)$ e um conjunto não-vazio de vértices $U \subseteq N_G(v)$ e emprega esse procedimento para gerar uma possível solução H.

De acordo com a Proposição 2.2, se existe uma raiz quadrada de G sem $\{C_3, C_5\}$ na qual a vizinhança de v é U, ela corresponde ao grafo H construído pelo procedimento. Mas a raiz desejada pode não existir, então é necessário verificar se H realmente é uma raiz quadrada de G. Além disso, H é livre de $\{C_3, C_5\}$, mas, como nos interessam raízes com cintura pelo menos 6, é necessário checar se é também livre de C_4.[1] O procedimento checa-solucao(G, H), que corresponde ao Algoritmo 5 apresentado no Capítulo 3, implementa esse teste.

[1]Uma solução não pode conter C_4 algum, seja ou não induzido. Mas note que, como H é livre de C_3, todo C_4 é induzido.

Algoritmo 1: raiz-quadrada-de-cintura-seis-com-vizinhanca

Entrada: Um grafo conexo G; um vértice v de G; um conjunto não vazio $U \subseteq N_G(v)$

Saída: Um grafo sem $\{C_3, C_5\}$ H tal que $H^2 = G$, se existir; "Não Tem Registro", caso contrário

```
1  Q ← fila vazia
2  H ← grafo vazio
3  para cada u ∈ V(G)
4  |  u.pai ← NULL
5  para cada u ∈ U
6  |  adicione uv a H
7  |  adicione u a Q
8  |  u.pai ← v
9  enquanto Q não está vazia
10 |  u ← Q.frente()
11 |  W ← N_G(u) ∩ (N_G[pai(u)] − N_H(pai(u)))
12 |  para cada w ∈ W
13 |  |  adicione uw a H
14 |  |  se w.pai = NULL
15 |  |  |  adicione w a Q
16 |  |  |  w.pai ← u
17 se checa-solucao(G, H)
18 |  devolve H
19 se não
20 |  devolve "Não Tem Registro"
```

As linhas 1 a 4 inicializam algumas estruturas. O laço nas linhas 5 a 8 constrói a vizinhança de v na raiz H como U. Cada vizinho adicionado a H é posto na fila Q e tem o vértice sendo processado (nesse caso, v) atribuído como seu pai. Atribuir um pai a um vértice serve tanto para marcar que esse vértice já foi adicionado à fila quanto para registrar um vizinho já conhecido dele em H. O laço seguinte (linhas 9 a 16) determina a vizinhança dos demais vértices na raiz usando a igualdade dada pela Proposição 2.2. A linha 17 invoca um procedimento para testar se H é uma solução.

Sejam m e n o número de arestas e vértices em G, respectivamente. Como cada vértice é adicionado à fila Q no máximo uma vez, as linhas 9 a 11 são executadas até n vezes e o laço nas linhas 12 a 16 é executado até m vezes. A linha 11 executa em tempo $O(n)$ e, como mostraremos no Capítulo 3, a linha 17 executa em $O(n^2)$. As demais linhas executam em tempo $O(1)$. Então o Algoritmo 1 tem complexidade de tempo $O(n^2)$.

Esse algoritmo requer a vizinhança de um vértice v em uma raiz quadrada para computar o restante da raiz. Se cada subconjunto não vazio de $N_G(v)$ fosse testado como vizinhança de v em alguma raiz, haveria $2^{|N_G(v)|} - 1$ candidatos. Porém, a Proposição 2.1 implica que se H tem cintura pelo menos 6, $G = H^2$, $uv \in E(H)$ e A é um componente de $G[N_G(u) \cap N_G(v)]$, então ou $N_H(v) = V(A) \cup \{u\}$ ou $N_H(u) = V(A) \cup \{v\}$. O Algoritmo 2 recebe como entrada um grafo G e uma aresta uv de G e tenta gerar uma solução que contenha essa aresta invocando o Algoritmo 1 para esses dois casos.

Algoritmo 2: raiz-quadrada-de-cintura-seis-com-aresta

Entrada: Um grafo conexo G com pelo menos três vértices; uma aresta uv de G

Saída: Um grafo H com cintura pelo menos 6 tal que $uv \in E(H)$ e $H^2 = G$, se existir; "Não Tem Registro", caso contrário

```
1  C ← N_G(u) ∩ N_G(v)
2  Calcule G[C]
3  se G[C] tem um ou dois componentes
4  |  A ← um componente (não vazio) de G[C]
5  |  H ← raiz-quadrada-de-cintura-seis-com-vizinhanca(G, v, {u} ∪ V(A))
6  |  se H ≠ "Não Tem Registro"
7  |  |  devolve H
8  |  H ← raiz-quadrada-de-cintura-seis-com-vizinhanca(G, u, {v} ∪ V(A))
9  |  devolve H
10 se não
11 |  devolve "Não Tem Registro"
```

Seja n o número de vértices em G. As linhas 1 e 4 executam em tempo $O(n)$. As linhas 2, 3, 5 e 8 têm custo $O(n^2)$, e as demais linhas custo $O(1)$. Portanto, o Algoritmo 2 tem complexidade de tempo $O(n^2)$.

Se um grafo conexo tem um ou dois vértices, ele é uma raiz quadrada dele mesmo. Se não, para computar uma raiz quadrada com cintura pelo menos 6, basta tentar encontrá-la com o Algoritmo 2 para cada aresta incidente em um vértice. Por eficiência, no Algoritmo 3 é selecionado um vértice de grau mínimo. Dessa forma, sua complexidade é $O(\delta(G) \cdot n^2)$, onde n é o número de vértices de G.

Algoritmo 3: raiz-quadrada-de-cintura-seis

Entrada: Um grafo conexo G com pelo menos três vértices

Saída: Uma raiz H com cintura pelo menos 6 de G, se existir; "Não Tem Registro", caso contrário

```
1 se N(G) tem menos de três elementos
2 |   devolve G
3 v ← um vértice de G com grau mínimo
4 para cada u ∈ N_G(v)
5 |   H ← raiz-quadrada-de-cintura-seis-com-aresta(G, uv)
6 |   se H ≠ "Não Tem Registro"
7 |   |   devolve H
8 devolve "Não Tem Registro"
```

3 CHECAGEM DE UMA SOLUÇÃO

A análise em Farzad et al. (2009) conclui que a complexidade do Algoritmo 3 é $O(\delta(G) \cdot n^4)$. O fator $O(n^4)$ é atribuído ao tempo para testar se uma possível solução H tem um C_4, na linha 17 do Algoritmo 1. Além disso, essa análise considera que testar se $H^2 = G$ tem a complexidade de multiplicar duas matrizes $n \times n$ (atualmente, $O(n^{2.373})$ (Alman e Williams, 2021)). Neste capítulo mostramos que é possível combinar esses dois testes em um procedimento $O(n^2)$.

Esse procedimento é baseado em um algoritmo pertencente ao folclore (veja, por exemplo, Yuster e Zwick (1997)) para detecção de C_4 (induzido ou não) em tempo quadrático, apresentado a seguir.

Algoritmo 4: livre-de-C_4

Entrada: Um grafo G

Saída: "Sim", se G é livre de C_4; "Não", caso contrário

```
1 M ← uma matriz indexada por V(G) × V(G) inicializada em 0
2 para cada v ∈ V(G)
3 |   para cada uw ∈ (N_G(v) 2)
4 |   |   se M[u, w] = 1
5 |   |   |   devolve "NÃO"
6 |   |   M[u, w] ← M[w, u] ← 1
7 devolve "SIM"
```

A ideia do Algoritmo 4 é muito simples: $M[u, v]$ conta o número de vizinhos comuns de u e v. Se há mais de um, então há um C_4 formado por u, v e dois de seus vizinhos comuns. Observe que, se G é livre tanto de C_4 quanto de C_3, então, ao final desse algoritmo, $M[u, v] = 1$ se e somente se $d_G(u, v) = 2$. Nesse caso, basta adicionar 1 também nas posições em que u e v são vizinhos para se construir a matriz de adjacência do quadrado do grafo. O Algoritmo 5 tira proveito disso para combinar os testes de se $H^2 = G$ e se H não contém C_4.

Algoritmo 5: checa-solucao-melhorado

Entrada: Um grafo G; um grafo H sem $\{C_3, C_5\}$

Saída: "Sim", se H não tem C_4 e $H^2 = G$; "Não", caso contrário

```
1  M ← matriz de adjacência de H
2  para cada v ∈ V(G)
3  |  para cada uw ∈ (N_G(v) 2)
4  |  |  se M[u, w] = 1
5  |  |  |  devolve "NÃO"
6  |  |  se não
7  |  |  |  M[u, w] ← M[w, u] ← 1
8  se M é matriz de adjacência de G
9  |  devolve "SIM"
10 se não
11 |  devolve "NÃO"
```

Note que M é inicializada como a matriz de adjacência de H em vez de em 0. Como H não tem C_3, nenhum 1 colocado na inicialização será lido na linha 4. Se a linha 8 for alcançada é porque H é livre de C_4, então basta checar se M é a matriz de adjacência de G para confirmar a validade da solução.

Seja n o número de vértices de G. A linha 2 é executada no máximo n vezes e, como encerram o procedimento se uma posição da matriz for visitada duas vezes, as linhas 3 a 7 são executadas no máximo $n^2 + 1$ vezes. Exceto as linhas 1 e 8, que executam em tempo $O(n^2)$, cada passo do algoritmo executa em $O(1)$. Então o Algoritmo 5 tem complexidade $O(n^2)$. Desse modo, obtemos

Teorema 3.1. *É possível resolver o Problema da Raiz Quadrada com Cintura pelo menos* 6 *em tempo* $O(\delta \cdot n^2)$. □

4 RAIZ QUADRADA COM CINTURA PELO MENOS SETE

O fator $O(\delta(G))$ na complexidade do Algoritmo 3 vem da necessidade de testar, para cada aresta incidente em um vértice, se há uma raiz quadrada com cintura pelo menos 6 que a contém. Uma forma de reduzir a complexidade do algoritmo é limitar o número de arestas que precisam ser testadas a uma constante, desde que o custo para isso seja inferior a $O(\delta(G) \cdot n^2)$.

A seguinte poposição sugere um método para, dada uma aresta que não está em nenhuma raiz quadrada com cintura 7, determinar uma aresta que esteja em uma raiz desse tipo, se houver, em tempo $O(n^2)$.

Proposição 4.1. *Sejam H um grafo com cintura pelo menos 7 e $G = H^2$. Se $uv \in E(G)$, mas $uv \notin E(H)$, então u e v tem um único vizinho comum w em H e $N_G[u] \cap N_G[v] = N_H[w]$.*

Demonstração. Sejam H, G, u e v como enunciados. Como $uv \in E(G) - E(H)$, deve existir um vizinho w comum a u e v em H. Além disso, não pode haver outro vizinho comum, ou H teria um C_4 e sua cintura não seria 7.

Todo vértice em $N_H[w]$ tem distância no máximo 2 até u e v em H, portanto $N_H[w] \subseteq N_G[u] \cap N_G[v]$. Se existisse um vértice a em $N_G[u] \cap N_G[v] - N_H[w]$, haveria um ciclo de comprimento $l \leq d_H(u,v) + d_H(v,a) + d_H(a,u) = 6$ em H. Então $N_G[u] \cap N_G[v] \subseteq N_H[w]$ e, consequentemente, $N_G[u] \cap N_G[v] = N_H[w]$. □

Corolário 4.2. *Sejam H um grafo com cintura pelo menos 7 tal que $G = H^2$ não é completo, v um vértice com grau máximo em G, u um vizinho de v em G mas não em H e w o vizinho comum a u e v em H. Então, para todo $x \in N_H[w] - \{v\}$, temos que $N_G[v] \cap N_G[x] \neq N_H[w]$ se e somente se $x = w$.*

Demonstração. Sejam H, G, v, u e w como enunciados. Todo vértice em $N_H(w) - \{v\}$ é um vizinho de v em G mas não em H. Então, de acordo com a Proposição 4.1, para qualquer x em $N_H(w) - \{v\}$, tem-se que $N_G[v] \cap N_G[x] = N_H[w]$. O vértice w não é o único elemento em $N_H(v)$. Caso contrário, teriamos que $N_G[v] = N_H[w]$ e, como v tem grau máximo em G, G seria completo. Seja a um vértice em $N_H(v) - \{w\}$. Tem-se que $a \notin N_H[w]$, caso contrário H teria um C_3. Mas $a \in N_G[w]$, então $N_G[v] \cap N_G[w] \neq N_H[w]$. □

O Algoritmo 6 recebe como entrada um grafo conexo G e calcula uma raiz quadrada com cintura pelo menos 7, se existir. Ele toma um vértice v de grau máximo e executa o Algoritmo 2 para procurar uma solução com uma aresta uv qualquer. Note que se G for completo, uma solução é um grafo estrela com o mesmo conjunto de vértices, então será encontrada nesse ponto. Se não for encontrada uma solução, busca-se um vizinho w de v tal que $N_G[v] \cap N_G[w] \neq N_G[v] \cap N_G[u]$ e chama-se o Algoritmo 2 para buscar uma solução com a aresta vw. De acordo com o Corolário 4.2, se existe uma solução, ela contém vw. Se esse

vizinho não existe, então não há solução. Note que o Algoritmo 2 pode retornar um grafo com cintura 6, então é necessário checar se a solução encontrada é livre de C_6. Veja Yuster e Zwick (1997) para um algoritmo que realiza essa checagem em tempo quadrático.

Algoritmo 6: raiz-quadrada-de-cintura-sete

Entrada: Um grafo conexo G

Saída: Uma raiz quadrada de G com cintura pelo menos 7, se existir; "Não Tem Registro", caso contrário

```
1  v ← um vértice de G com grau máximo
2  u ← um vizinho de v
3  H ← raiz-quadrada-de-cintura-seis-com-aresta(G, uv)
4  se H = "Não Tem Registro"
5  |  C ← N_G[v] ∩ N_G[u]
6  |  para cada w ∈ C
7  |  |  se N_G[v] ∩ N_G[w] ≠ C
8  |  |  |  H ← raiz-quadrada-de-cintura-seis-com-aresta(G, vw)
9  |  |  |  encerra o laço
10 se H ≠ "Não Tem Registro" e H é livre de C_6
11 |  devolve H
12 se não
13 |  devolve "Não Tem Registro"
```

Seja n o número de vértices de G. As linhas 6 e 7 executam $O(n)$ vezes. As linhas 1, 3, 8 e 10 executam em tempo $O(n^2)$. As linhas 5 e 7 executam em $O(n)$. As demais linhas executam em $O(1)$. Então o Algoritmo 2 tem complexidade de tempo $O(n^2)$. Desse modo, obtemos

Teorema 4.3. *É possível resolver o Problema da Raiz Quadrada com Cintura pelo menos* 7 *em tempo* $O(n^2)$.

5 CONSIDERAÇÕES FINAIS

A modificação ao algoritmo para encontrar raízes quadradas com cintura 6 já foi apresentada na 10th Latin American Workshop on Cliques in Graphs (LAWCG '22). Um resumo deste trabalho, incluindo o algoritmo para raízes com cintura 7, foi submetido para publicação na revista Matemática Contemporânea.

Uma questão interessante é como a complexidade de tempo do Problema da Raiz Quadrada com Cintura pelo menos k varia em função de $k \geq 6$. Embora uma solução quadrática tenha sido atingida apenas para encontrar uma raiz quadrada com cintura pelo menos 7, acreditamos que seja possível melhorar a complexidade também para raízes com cintura pelo menos 6. Uma extensão direta da solução apresentada aqui para $k > 7$ seria buscar uma raiz com o Algoritmo 6 (raiz-quadrada-de-cintura-sete) e testá-la para ciclos de tamanho de 7 até k. Mas a detecção de ciclos de tamanho ímpar se torna um gargalo e a complexidade $O(n^2)$ não é atingida.

De fato, checar a validade da raiz construída é o gargalo também para $k \in \{6, 7\}$. Com esse gargalo em mente, buscamos manter os algoritmos e as discussões de suas complexidades simples em vez de tomarmos os limites mais justos para cada passo. Mas é possível modificar o Algoritmo 1 (raiz-quadrada-de-cintura-seis-com-vizinhanca) para construir H em tempo $O(m)$ em vez de $O(n^2)$. Soluções que dispensem esses testes podem ter complexidades menores, como os algoritmos para encontrar raízes acíclicas ($k = \infty$) que atingem uma complexidade $O(m + n)$.

REFERÊNCIAS

Adamaszek, A. e Adamaszek, M. (2011). Uniqueness of graph square roots of girth six. *The Electronic Journal of Combinatorics [electronic only]*, 18(1):Research Paper P139, 5 p., electronic only–Research Paper P139, 5 p., electronic only.

Alman, J. e Williams, V. V. (2021). A refined laser method and faster matrix multiplication. Em *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, páginas 522–539. SIAM.

Andreae, T. (1995). On the traveling salesman problem restricted to inputs satisfying a relaxed triangle inequality. *Networks: An International Journal*, 38(2):59–67.

Andreae, T. e Bandelt, H.-J. (1995). Performance guarantees for approximation algorithms depending on parametrized triangle inequalities. *SIAM Journal on Discrete Mathematics*, 8(1):1–16.

Bender, M. A. e Chekuri, C. (2000). Performance guarantees for the tsp with a parameterized triangle inequality. *Information Processing Letters*, 73(1):17–21.

Farzad, B., Lau, L. C., Le, V. B. e Tuy, N. N. (2009). Computing Graph Roots Without Short Cycles. Em Albers, S. e Marion, J.-Y., editores, *26th International Symposium on Theoretical Aspects of Computer Science STACS 2009*, Proceedings of the 26th Annual Symposium on the Theoretical Aspects of Computer Science, páginas 397–408, Freiburg, Germany. IBFI Schloss Dagstuhl.

Fleischner, H. (1974). The square of every two-connected graph is hamiltonian. *Journal of Combinatorial Theory, Series B*, 16(1):29–34.

Karimi, M. (2013). Square root finding in graphs. Master of science.

Lau, H. T. (1980). *Finding a Hamiltonian Cycle in the Square of a Block*. Tese de doutorado, School of Computer Science, McGill University, Montreal - Canadá.

Lin, G.-H., Kearney, P. E. e Jiang, T. (2000). Phylogenetic k-root and steiner k-root. Em *Algorithms and Computation: 11th International Conference*, páginas 539–551, Taipei - Taiwan.

Lin, Y.-L. e Skiena, S. S. (1991). Algorithms for square roots of graphs. Relatório técnico, Department of Computer Science, Stony Brook.

Motwani, R. e Sudan, M. (1994). Computing roots of graphs is hard. *Discrete Applied Mathematics*, 54(1):81–88.

Nishimura, N., Ragde, P. e Thilikos, D. M. (2002). On graph powers for leaf-labeled trees. *Journal of Algorithms*, 42(1):69–108.

Ross, I. C. e Harary, F. (1960). The square of a tree. *The Bell System Technical Journal*, 39(3):641–647.

Sejanina, M. (1960). On an ordering of the set of vertices of a connected graph. Relatório técnico, Publ. Fac. Sci. Univ. Brno.

Yuster, R. e Zwick, U. (1997). Finding even cycles even faster. *SIAM Journal on Discrete Mathematics*, 10(2):209–222.