Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 8, n. 1, 2021.

Trabalho apresentado no XL CNMAC, Evento Virtual - Co-organizado pela Universidade do Mato Grosso do Sul (UFMS).

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Algoritmos lineares para emparelhamentos conexos máximos ponderados e não ponderados

Bruno P. Masquio[1]
UERJ, Rio de Janeiro, RJ
Paulo E. D. Pinto[2]
UERJ, Rio de Janeiro, RJ
Jayme L. Szwarcfiter[3]
UFRJ/UERJ, Rio de Janeiro, RJ

Resumo. Problemas de emparelhamentos em grafos são bem conhecidos e estudados, nos quais desejamos determinar conjuntos de arestas não adjacentes duas a duas. Surgiu um interesse pelo estudo do emparelhamento cujo subgrafo induzido pelos vértices do emparelhamento é conexo, o qual pode ser estudado em grafos poderados e não ponderados. Para os não ponderados, deseja-se obter um emparelhamento de cardinalidade máxima. Esse problema já foi identificado polinomial, porém sua complexidade exata era desconhecida. Obtivemos um algoritmo linear que, a partir de um emparelhamento máximo, resolve o problema para grafos gerais. Já para os ponderados, ainda não se conhece sua complexidade para grafos gerais, porém apresentamos um algoritmo linear para resolvê-lo para árvores.

Palavras-chave. Algoritmos. Emparelhamento. Conexo. Ponderado. Árvores.

1 Introdução

A área de estudo de emparelhamentos em grafos é bem estabelecida e possui resultados expressivos. Recentemente, surgiu o interesse de estudar emparelhamentos restritos a subgrafos [1] [2]. Nessa abordagem, procura-se resolver o problema preservando certas propriedades de subgrafos induzidos pelos vértices do emparelhamento. Algumas dessas propriedades, por exemplo, são que esses subgrafos sejam 1-regulares, acíclicos, desconexos ou conexos.

Apresentaremos algumas definições e notações utilizadas. Considere G um grafo e M um emparelhamento de G. Denotamos por $G[M]$, o subgrafo induzido de G pelos vértices incidentes às arestas de M. Seja v um vértice de G. Se $G - v$ possui mais componentes conexas que G, então v é dita como uma articulação. Denotamos por $N(v)$ o conjunto de vértices vizinhos de v em G. Se existe uma aresta de M incidente a v, então v é M-saturado. Um emparelhamento M é dito *conexo* se $G[M]$ é conexo. Usamos $\beta(G)$ e $\beta_c(G)$ para denotar, respectivamente, as cardinalidades de um emparelhamento máximo e de um emparelhamento conexo máximo, ambos não ponderados.

Neste artigo, vamos tratar dos problemas de emparelhamentos conexos máximos não ponderados e ponderados em um grafo G. No primeiro caso, determina-se um emparelhamento conexo M de cardinalidade máxima. Já no segundo, encontra-se um emparelhamento conexo M tal que a soma dos pesos de suas arestas seja máxima. Neste trabalho, quando não for explicitado, o emparelhamento considerado é não ponderado.

[1]brunomasquio@ime.uerj.br
[2]pauloedp@ime.uerj.br
[3]jayme@nce.ufrj.br. Projeto parcialmente financiado por FAPERJ

(a) Um emparelhamento não ponderado (b) Um emparelhamento ponderado

Figura 1: Um grafo e diferentes tipos de emparelhamentos conexos máximos

Note que, em um mesmo grafo, nem sempre são iguais as cardinalidades de um emparelhamento conexo máximo ponderado e de um emparelhamento conexo máximo não ponderado, o que exemplificamos na Figura 1. O emparelhamento da esquerda tem cardinalidade 2, enquanto o da direita, 1. Diante disso, esses problemas têm um tratamento diferente.

2 Emparelhamento conexo máximo não ponderado

Nesta seção, vamos apresentar um algoritmo linear para obter um emparelhamento conexo máximo não ponderado em tempo linear, dado um emparelhamento máximo geral. A seguir, mostramos um bom resultado já obtido que relaciona as cardinalidades dos emparelhamentos máximos e dos emparelhamentos conexos máximos.

Teorema 2.1. *Se G é conexo, então $\beta_c(G) = \beta(G)$ [1].*

Embora os autores do Teorema 2.1 tenham apresentado um bom resultado para o problema, não se conhecia um algoritmo para encontrar um emparelhamento conexo máximo e nem era evidente a complexidade exata desse procedimento. Nossa contribuição é um algoritmo linear para obter um emparelhamento conexo máximo, a partir de um emparelhamento máximo qualquer.

A ideia do algoritmo, baseada no Teorema 2.1, é que, dado um emparelhamento máximo M, é possível construir um emparelhamento conexo de mesma cardinalidade apenas alterando arestas de M, sem alterar a sua cardinalidade.

Vamos descrever como a alteração de arestas pode ser feita. Seja um emparelhamento máximo M tal que $G[M]$ seja desconexo e r um vértice qualquer M-saturado. Considere C_r a componente de $G[M]$ que contem r e $W = \{w | w \in N(v) \setminus V(C_r), \forall v \in C_r\}$, ou seja, W é o conjunto de vértices fora da componente C_r que são vizinhos de algum vértice dessa componente. Podemos observar que, em G, existe um vértice saturado $u \notin C_r$ adjacente a algum vértice w de W. Logo, é possível aumentar a cardinalidade de C_r removendo de M a aresta saturada de u e adicionando a aresta (u, w). Esse procedimento pode ser feito até que $G[M]$ seja conexo.

O algoritmo utiliza duas filas, Q_s e Q_n para guardar vértices saturados e vértices não saturados por M, respectivamente, e um conjunto C, no qual são inseridos vértices de C_r ou novos vértices adicionados a essa componente. O loop principal inclui pelo menos um vértice a C, a cada iteração, e é dividido em dois outros loops auxiliares. O primeiro loop auxiliar, executa Expande(v) para cada vértice v de Q_s, analisa $N(v)$ e enfila adequadamente cada vértice dessa vizinhança ainda não enfilado. Já o segundo loop auxiliar, executa a função TentaConectar(v) para cada vértice v de Q_n. Caso exista $w \in N(v) \setminus C$, então w está saturado por uma aresta (w, u) e é feita a operação de troca de arestas em M, removendo a aresta (w, u) e adicionando a aresta (v, w).

DOI: 10.5540/03.2021.008.01.0354

Algoritmo 1 Emparelhamento conexo máximo não ponderado para grafos gerais

```
1: função Expande(Vértice: v)
2:     para w ∈ N(v) \ C faça
3:         se w está M-saturado então
4:             C ← C ∪ {w}
5:             Insere w em Q_s
6:         senão se w ∉ W então
7:             W ← W ∪ {w}
8:             Insere w em Q_n
9:
10: função TentaConectar(Vértice: v)
11:     se ∃w ∈ N(v) \ C então
12:         u ← Vértice emparelhado com w em M
13:         M ← M \ {(u, w)} ∪ {(v, w)}
14:         C ← C ∪ {v, w}
15:         Insere v e w em Q_s
16:
17: #Entrada: G: Grafo, M: emparelhamento máximo em G
18: W ← ∅;   Q_s ← ∅;   Q_n ← ∅;
19: r ← Vértice qualquer M-saturado
20: C ← {r}
21: Insere r em Q_s
22: enquanto |C| < |M| * 2 faça
23:     enquanto Q_s não está vazia faça
24:         v ← Vértice retirado da frente de Q_s
25:         Expande(v)
26:     enquanto Q_n não está vazia faça
27:         v ← Vértice retirado da frente de Q_n
28:         TentaConectar(v)
29: retorna M
```

A Figura 2 ilustra dois emparelhamentos máximos para um grafo. O da direita, que é conexo, pode ser obtido a partir do Algoritmo 2 tendo como entrada o emparelhamento da esquerda, que é desconexo.

Teorema 2.2. *O Algoritmo 1 tem complexidade de tempo $O(n + m)$.*

Demonstração. Observe que o loop principal é executado até que a componente seja do tamanho desejado. No início de cada iteração, sabemos que $G[M]$ ainda não é conexo por conta da condição de parada do loop. Como, ao final do loop, pelo menos um vértice é adicionado a C, o número de iterações é $O(n)$. Considerando que os conjuntos C e W funcionam como marcações para as filas, cada vértice $v \in V(G)$ só pode ser processado, no máximo, uma vez nas funções Expande(v) e TentaConectar(v). Observe que a execução dessas funções analisa a vizinhança de v e, portanto, tem uma complexidade linear no grau de v. Logo, o algoritmo completo tem complexidade linear em relação aos vértices e às arestas. □

Note que para usar o Algoritmo 1, é necessário calcular o emparelhamento máximo previamente. Para a obtenção de um emparelhamento máximo, é conhecido um algoritmo $O(m\sqrt{n})$ [3] para grafos em geral. Logo, essa complexidade também vale para um emparelhamento conexo

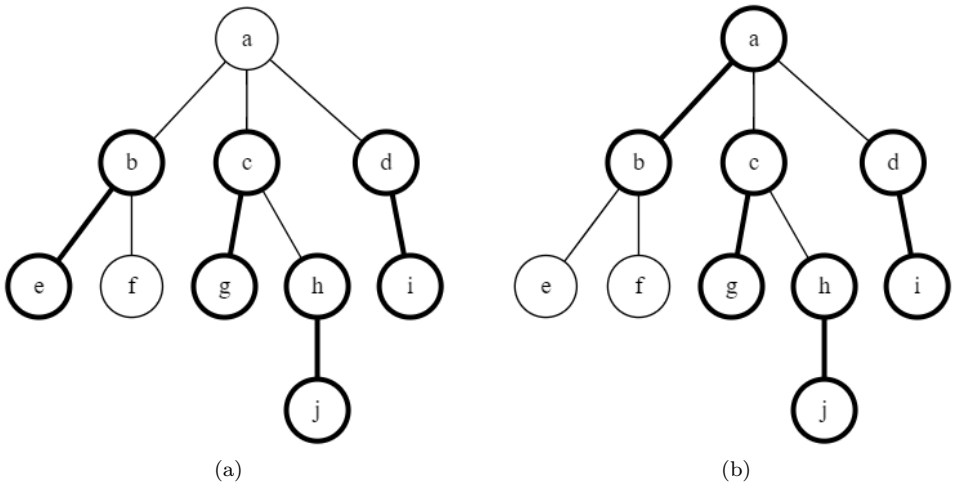

Figura 2: Emparelhamentos desconexo em (a) e conexo em (b)

máximo. Ou seja, o problema de emparelhamentos conexos máximos tem solução polinomial e restrita à complexidade de emparelhamentos máximos. Observe que são conhecidas algumas classes, como árvores e grafos bloco, cujos emparelhamentos máximos podem ser calculados em tempo linear [2] [4]. Para essas classes, portanto, a existência do Algoritmo 1 mostra que o problema do emparelhamento conexo máximo tem complexidade linear também.

3 Emparelhamento conexo máximo ponderado para árvores

O problema de emparelhamentos conexos máximos ponderados ainda não possui resultados relevantes em relação a grafos gerais, pois não se conhece a sua complexidade. Nossa contribuição neste artigo é um algoritmo de complexidade linear, ao restrigir o problema para árvores. Vamos descrevê-lo a seguir, juntamente com sua base teórica.

As seguintes notações serão utilizadas nesta seção. Seja T uma árvore e dois vértices $r, v \in V(T)$. Denotamos T^r como a árvore T enraizada em r, e T^r_v como a subárvore de T^r enraizada em v.

O seguinte teorema se aplica a grafos em geral ponderados ou não. Apesar disso, sua aplicação neste artigo é somente para árvores.

Teorema 3.1. *Seja G um grafo conexo e M um emparelhamento conexo máximo ponderado. Então, M satura todas as articulações.*

Demonstração. Por absurdo, suponha que v é uma articulação de G que não está saturada por M. Considere os elementos do conjunto $C = \{C_1, C_2, \ldots, C_s\}$ como as componentes conexas de $S = G - v$. Sabemos que, por v ser uma articulação, $|C| \geq 2$. Note que as arestas de M têm que pertencer a somente uma componente C_i de S, pois, caso contrário, $G[M]$ seria desconexo. Vamos mostrar que é possível aumentar M. Considere a componente $C_j \in S$, $i \neq j$. Seja $U = V(C_i) \cap N(v)$ e $W = V(C_j) \cap N(v)$. Vamos considerar dois casos. No primeiro, existe um vértice $u \in U$ não saturado. Observe que v é adjacente a algum vértice saturado, pois, caso contrário, seria possível adicionar a M, alternadamente, as arestas do caminho de v até um vértice saturado. Portanto, $M \cup (u, v)$ ainda seria conexo. No segundo caso, todos os vértices de U estão M-saturados. Ainda assim, é possível adicionar a M a aresta (v, w), $w \in W$. Em ambos os casos, portanto, M não é máximo, absurdo, o que implica que v deve estar saturado em qualquer emparelhamento conexo máximo ponderado. □

DOI: 10.5540/03.2021.008.01.0354

Pelo Teorema 3.1, sabemos que, sem perder a generalidade, um emparelhamento máximo ponderado em uma árvore satura todas as articulações também. A partir dessa informação, é possível definir uma ideia de programação dinâmica, que leva a um algoritmo linear para determinar emparelhamentos conexos máximos ponderados em árvores. A ideia é que, sabendo que cada articulação v deve estar saturada, deve-se decidir quais vizinhos de v maximizam a soma dos pesos das arestas de um emparelhamento em T_v^r, sendo r um vértice qualquer escolhido inicialmente como a raiz.

Considere a árvore T^r, M o emparelhamento conexo máximo ponderado de T^r e $v \neq r$ um vértice de T^r, que seja articulação em T_r. Observe que, em M restrito a T_v^r, v poderia estar emparelhado ou não. Neste último caso, ele estaria emparelhado com o seu pai em T^r. Definimos a soma dos pesos das arestas de um emparelhamento máximo ponderado em T_v^r como B_v se v estiver emparelhado com um dos seus filhos, e $\overline{B_v}$ se v estiver emparelhado com seu pai. Além disso, considere $A(r, v)$ como o conjunto de vértices filhos de v em T_v^r e $w(v, u)$ o peso da aresta (v, u). Logo, podemos chegar às seguintes fórmulas.

$$\overline{B_v} = \sum_{u \in A(r,v)} B_u$$

$$B_v = max\left(\overline{B_a} + w(a, v) + \sum_{u \in A(r,v)\setminus\{a\}} B_u\right), \forall a \in A(r, v)$$

Mediante a essa formulação, a solução do problema consiste em calcular B_r.

O algoritmo descrito a seguir constrói, dinamicamente, um emparelhamento conexo máximo M da seguinte forma. A partir de uma articulação arbitrária r eleita como raiz, são feitas duas buscas. A primeira processa os vértices das folhas à r. Obtém, para cada vértice u, um vértice filho de u que maximiza B_u, o qual é representado por s_u no algoritmo. Além disso é calculado $\overline{B_u}$ a partir da soma de B_w para todos os seus filhos w. A segunda busca é responsável por construir M, processando os vértices de r às folhas de modo que, quando se processa o vértice u, caso u não faça parte de M, adicionamos (s_u, u) a M. Ao final, M será um emparelhamento conexo máximo.

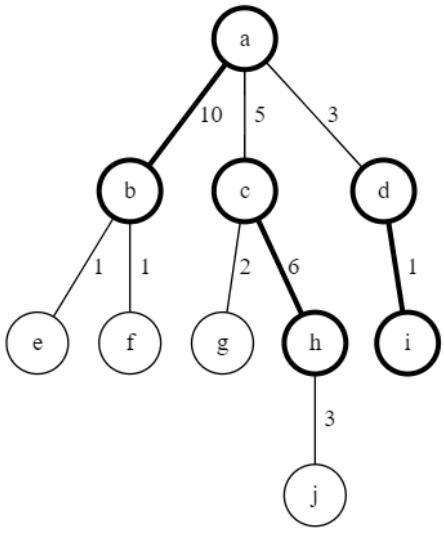

Figura 3: Um emparelhamento conexo máximo ponderado

DOI: 10.5540/03.2021.008.01.0354

Algoritmo 2 Emparelhamento conexo máximo ponderado para árvores

função BUSCA1(Vértice: u, Vértice: v)
    para w vizinho de v **faça**
        se $w \neq u$ **então**
            BUSCA1(v, w)
            $\overline{B_v} \leftarrow \overline{B_v} + B_w$
    para w vizinho de v **faça**
        se $w \neq u$ **então**
            $Btemp \leftarrow \overline{B_v} - B_w + \overline{B_w} + w(v, w)$
            se $Btemp > B_v$ **então**
                $B_v \leftarrow Btemp$
                $s_v \leftarrow w$

função BUSCA2(Vértice: u, Vértice: v)
    se v é articulação e não está M-saturada **então**
        $M \leftarrow M \cup \{(v, s_v)\}$
    para w vizinho de v **faça**
        se $w \neq u$ **então**
            BUSCA2(v, w)

#Entrada: T: Árvore ponderada
#B_v: Inteiro, $\forall v \in V(T)$
#$\overline{B_v}$: Inteiro, $\forall v \in V(T)$
#s_v: Vértice, $\forall v \in V(T)$
para v em $V(T)$ **faça**
    $B_v \leftarrow 0$; $\overline{B_v} \leftarrow 0$;
$r \leftarrow$ Articulação qualquer de T
BUSCA1(r, r)
BUSCA2(r, r)
Retorna M

A Figura 3 ilustra uma árvore e seu emparelhamento conexo máximo ponderado de peso 17. Esse emparelhamento foi obtido com a execução do Algoritmo 2. A Tabela 1 mostra o valor das variáveis B_v, $\overline{B_v}$ e s_v obtidas após a BUSCA1(a, a), sendo a articulação a escolhida como raiz. Observe que trata-se da mesma árvore da Figura 2 e que a cardinalidade do emparelhamento ponderado é menor do que a do não ponderado.

Tabela 1: Exemplo das variáveis obtidas a partir de uma execução do Algoritmo 2

v	B_v	$\overline{B_v}$	s_v	v	B_v	$\overline{B_v}$	s_v
a	17	8	b	f	0	0	–
b	1	0	f	g	0	0	–
c	6	3	h	h	3	0	j
d	1	0	i	i	0	0	–
e	0	0	–	j	0	0	–

DOI: 10.5540/03.2021.008.01.0354

Teorema 3.2. *O Algoritmo 2 pode ser implementado em complexidade de tempo $O(n)$.*

Demonstração. A parte central do algoritmo é dividida nas buscas BUSCA1(r,r) e BUSCA2(r,r). Observe que o percurso de ambas as buscas é de uma busca em profundidade. Então, em cada busca, todos os vértices são processados uma única vez e, portanto, cada aresta é visitada exatamente 2 vezes. Todas as operações restantes de conjuntos podem ser feitas em tempo constante a partir de uma lista de acesso direto. Portanto, a execução completa Algoritmo 2 tem complexidade linear em relação aos vértices. □

4 Conclusões

Neste artigo, discutimos os problemas de emparelhamentos conexos máximos ponderados e não ponderados. Contribuimos com dois novos algoritmos lineares.

O Algoritmo 1 mostra que é possível calcular um emparelhamento conexo máximo não ponderado em um grafo geral em tempo linear se já se conhece um emparelhamento máximo não ponderado. Já o Algoritmo 2 mostra que calcular um emparelhamento conexo máximo ponderado pode ser feito em tempo linear para árvores.

Evidenciamos que o problema de determinar emparelhamentos conexos máximos não ponderados está resolvido e tem solução polinomial. Em particular, mostramos um algoritmo de complexidade linear, se for dado um emparelhamento máximo qualquer do grafo. Em contraste, nenhuma estratégia foi formulada para a solução do problema de emparelhamentos conexos máximos ponderados para grafos em geral. A contribuição dada neste artigo é uma solução linear para o problema, quando restrito à classe de árvores.

Assim, algumas questões ainda ficam em aberto para trabalhos futuros:

- Qual a complexidade do problema de emparelhamentos conexos máximos ponderados para grafos gerais?
- Existem outras classes de grafos para as quais é possível calcular um emparelhamento conexo máximo ponderado em tempo polinomial?

Referências

[1] Goddard W., Hedetniemi S. M. , Hedetniemi S. T. and Laskar R. Generalized Subgraph-restricted Matchings in Graphs, *Discrete Mathematics*, volume 293, issues 1–3, pages 129-138, 2005. DOI: 10.1016/j.disc.2004.08.027.

[2] Masquio, B. P. Emparelhamentos desconexos, Dissertação de Mestrado, UERJ, 2019.

[3] Micali, S. and Vazirani, V. V. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs, *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 17-27, 1980. DOI: 10.1109/SFCS.1980.12.

[4] Savage, C. Maximum matchings and trees, *Information Processing Letters*, volume 10, issues 4–5, pages 202-205, 1980. DOI: 10.1016/0020-0190(80)90140-4.

