LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2020

Algoritmos lineares para emparelhamentos conexos

Bruno P. Masquio
Universidade do Estado do Rio de Janeiro
São Francisco Xavier, 524 - Maracanã, Rio de Janeiro/RJ
brunomasquio@ime.uerj.br

Paulo E. D. Pinto
Universidade do Estado do Rio de Janeiro
São Francisco Xavier, 524 - Maracanã, Rio de Janeiro/RJ
pauloedp@ime.uerj.br

Jayme L. Szwarcfiter
Universidade Federal do Rio de Janeiro e Universidade do Estado do Rio de Janeiro
Av. Pedro Calmon, 550 - Cidade Universitária da Universidade Federal do Rio de Janeiro, Rio de Janeiro/RJ
jayme@nce.ufrj.br

RESUMO

Problemas de emparelhamentos em grafos são bem conhecidos e estudados, nos quais desejamos determinar conjuntos de arestas não adjacentes duas a duas. Este trabalho se dedica ao estudo do emparelhamento cujos subgrafos induzidos pelos vértices do emparelhamento são conexos ou desconexos. Podemos considerar também grafos poderados e não ponderados. Para os não ponderados, deseja-se obter um emparelhamento de cardinalidade máxima, enquanto que para os ponderados o objetivo é determinar um emparelhamento cuja soma dos pesos das arestas do emparelhamento é máxima. Enquanto conjecturamos que o problema de emparelhamento desconexo máximo é NP-completo, o problema de emparelhamento conexo máximo é polinomial. Mostramos um algoritmo linear que, a partir de um emparelhamento máximo, determina um emparelhamento conexo máximo. Para emparelhamento conexo ponderado, sua complexidade ainda é desconhecida para grafos gerais. Entretanto, apresentamos um algoritmo linear para resolvê-lo para árvores.

PALAVRAS CHAVE. Emparelhamento, Conexo, Ponderado.
Tópicos. Teoria e Algoritmos em Grafos

ABSTRACT

Graph matching problems are well known and studied, in which we want to find sets of pairwise non-adjacent edges. This paper is dedicated to the study of matchings in which the induced subgraphs by the vertices of matchings are connected or disconnected. We can also consider it for weighted or unweighted graphs. For the unweighted, we want to obtain a matching with the maximum cardinality, while for the weighted, we look for a matching whose sum of the matching edges weights is maximum. While we conjecture that the problem of maximum disconnected matching is NP-complete, the problem of maximum connected matching is polynomial. We show a linear algorithm that, having a maximum matching as input, determines a maximum connected matching. For maximum weighted connected matching, its complexity is still unknown for general graphs. Nevertheless, we present a linear algorithm to solve it for trees.

KEYWORDS. Matching. Connected. Weighted.
Paper topics. Graph theory and algorithms

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

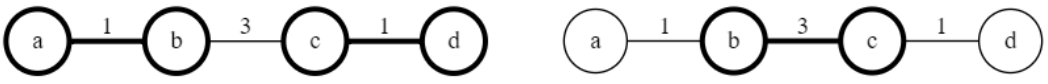
1. Introdução

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 (Goddard et al. [2005]; Masquio [2019]). 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. Um subconjunto $S \subset V(G)$ é dito um *separador* de G se existem dois vértices que estão na mesma componente conexa de G e em componentes conexas distintas de $G - S$. Também dizemos que S é um *separador minimal* de G se S é um separador e não há subconjunto próprio de S que também seja um separador. Se $|S| = 1$, então o vértice $s \in S$ é dito uma *articulação*. Se $G - S$ possuir mais de uma componente conexa com pelo menos dois vértices, então S é um *separador composto*. 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 e *desconexo* se $G[M]$ é desconexo. Usamos $\beta_c(G)$ e $\beta_d(G)$ para denotar, respectivamente, as cardinalidades de um emparelhamento conexo máximo e de um emparelhamento desconexo máximo, ambos não ponderados.

Neste artigo, vamos tratar dos problemas de emparelhamento conexo máximo não ponderado e ponderado 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. Ao final, vamos mostrar diferenças de tratamento e complexidades conhecidas dos problemas de emparelhamento conexo máximo e desconexo máximo.

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 $\{(a, b), (c, d)\}$ é um emparelhamento conexo máximo e tem cardinalidade 2, enquanto o da direita é um emparelhamento conexo máximo ponderado$\{(b, c)\}$, com cardinalidade igual a 1. Diante disso, podemos imaginar que esses problemas tenham tratamentos computacionais diferentes.



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

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

Note também que as cardinalidades de um emparelhamento conexo máximo e de um emparelhamento desconexo máximo podem ser iguais ou diferentes, dependendo do grafo. No grafo da Figura 2, o emparelhamento desconexo máximo $\{(a, b), (d, e)\}$ tem cardinalidade 2, enquanto o emparelhamento conexo máximo $\{(a, b), (c, d), (e, f)\}$ tem cardinalidade 3. Em contraste, observe que, no grafo Figura 3, os emparelhamentos desconexo máximo $\{(b, e), (c, g), (h, j), (d, i)\}$ e conexo máximo $\{(a, b), (c, g), (h, j), (d, i)\}$ possuem a mesma cardinalidade, igual a 4.

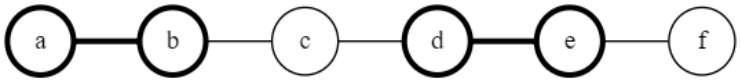


Figura 2: Um emparelhamento desconexo

2. Emparelhamento conexo máximo não ponderado

Nesta seção, vamos apresentar um algoritmo linear para obter um emparelhamento conexo máximo não ponderado de um grafo G, em tempo linear, dado um emparelhamento máximo qualquer.

Inicialmente, mostramos um resultado existente, que relaciona as cardinalidades do emparelhamento máximo e do emparelhamento conexo máximo.

Teorema 2.1. *Se G é conexo, então $\beta_c(G) = \beta(G)$ (Goddard et al. [2005])*

A prova do Teorema 2.1 é baseada no fato de que, em um grafo G, se M é um emparelhamento máximo, $G[M]$ é desconexo e C é uma componente conexa de $G[M]$, então é possível redefinir as arestas de M de forma a incrementar vértices em C, sucessivamente, até ficarmos com uma única componente. Isso é feito obtendo-se caminhos cujas arestas estão alternadamente em M e em $E(G) \setminus M$, construído da seguinte forma. Considere C uma componente conexa de $G[M]$ e $u \notin V(C)$ um vértice M-saturado. Considerando G conexo, existe um caminho mínimo P de um vértice qualquer $a \in V(C)$ até u. Considere b e c o segundo e terceiro vértices de P, respectivamente. Note que b não está M-saturado, c está M-saturado e $c \notin V(C)$. Então, em M, remove-se a aresta do emparelhamento incidente a c e adiciona-se a aresta (b, c).

Embora os autores do Teorema 2.1 tenham apresentado um resultado interessante 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 contém 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 adiciona adequadamente a fila cada vértice dessa vizinhança ainda não entrou na fila. 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).

Algoritmo 1 Emparelhamento conexo máximo não ponderado para grafos gerais

função EXPANDE(Vértice: v)
    para $w \in N(v) \setminus C$ **faça**
        se w está M-saturado **então**
            $C \leftarrow C \cup \{w\}$
            Insere w em Q_s
        senão se $w \notin W$ **então**
            $W \leftarrow W \cup \{w\}$
            Insere w em Q_n

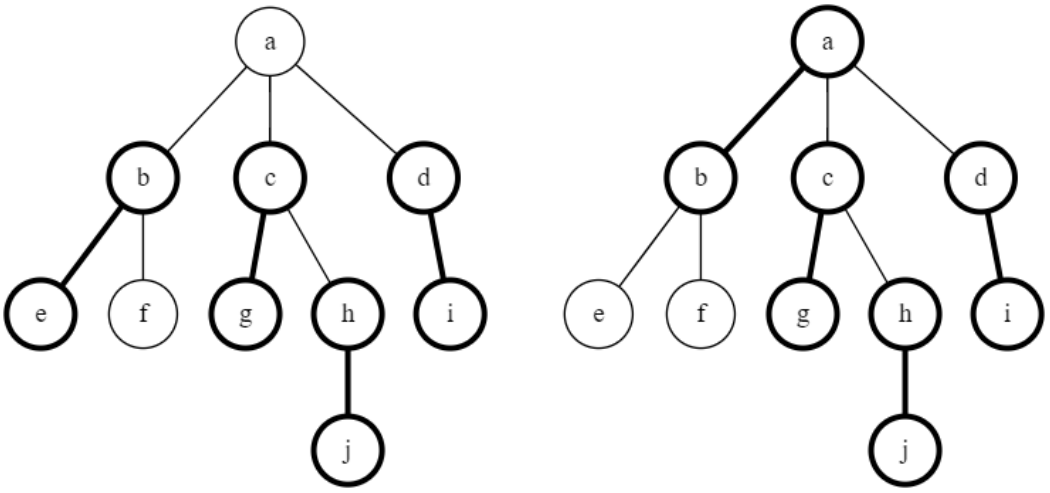
função TENTACONECTAR(Vértice: v)
    se $\exists w \in N(v) \setminus C$ **então**
        $u \leftarrow$ Vértice emparelhado com w em M
        $M \leftarrow M \setminus \{(u, w)\} \cup \{(v, w)\}$
        $C \leftarrow C \cup \{v, w\}$
        $W \leftarrow W \setminus \{v\}$
        Insere v e w em Q_s

#Entrada: G: Grafo, M: emparelhamento máximo em G
$W \leftarrow \emptyset$; $Q_s \leftarrow \emptyset$; $Q_n \leftarrow \emptyset$;
$r \leftarrow$ Vértice qualquer M-saturado
$C \leftarrow \{r\}$
Insere r em Q_s
enquanto $|C| < |M| * 2$ **faça**
    enquanto Q_s não está vazia **faça**
        $v \leftarrow$ Vértice retirado da frente de Q_s
        EXPANDE(v)
    enquanto Q_n não está vazia **faça**
        $v \leftarrow$ Vértice retirado da frente de Q_n
        TENTACONECTAR(v)
retorna M

A Figura 3 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. Esse emparelhamento pode ser obtido pelo Algoritmo 1 se escolhermos, por exemplo, c como o vértice inicial. Ele é colocado na fila Q_s e em C. O procedimento EXPANDE(c) adiciona os vértices g e h à fila Q_s e ao conjunto C. Adiciona, também, o vértice a à fila Q_n e ao conjunto W. Em seguida, o vértice a, da fila Q_n é processado pelo procedimento TENTACONECTAR(a). É encontrado o vértice $b \notin N(a) \setminus C$. Então, em M é retirada a aresta (b, e) e inserida a (a, b). Os vértices a e b são colocados em C. No restante do algoritmo mais nenhuma alteração em M é feita e o emparelhamento conexo máximo é obtido.

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



(a) (b)

Figura 3: Emparelhamentos desconexo em (a) e conexo em (b)

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 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 de complexidade $O(m\sqrt{n})$ (Micali e Vazirani [1980]) para grafos em geral. Logo, essa complexidade também vale para um emparelhamento conexo máximo. Ou seja, o problema de emparelhamento conexo máximo tem solução polinomial e restrita à complexidade de emparelhamento máximo. Observe que são conhecidas algumas classes, como árvores e grafos bloco, cujos emparelhamentos máximos podem ser calculados em tempo linear (Masquio [2019]; Savage [1980]). Para essas classes, portanto, a existência do Algoritmo 1 mostra que o problema do emparelhamento conexo máximo tem complexidade linear também.

A Tabela 2 mostra um exemplo de execução do Algoritmo 1 com o emparelhamento de entrada $M = \{(c, g), (h, j), (b, e), (d, i)\}$ e escolha do vértice c como o inicial. Cada linha representa o estado das variáveis após a execução da função especificada na primeira coluna. As linhas estão em ordem crescente de acordo com o momento em que são executadas.

3. Emparelhamento conexo máximo ponderado para árvores

O problema de emparelhamento conexo máximo ponderado 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.

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

Função executada	C	W	Q_s	Q_n	M
EXPANDE(c)	{c, g, h}	{a}	{g,h}	{a}	{(c,g), (h,j), (b,e), (d,i)}
EXPANDE(g)	{c, g, h}	{a}	{h}	{a}	{(c,g), (h,j), (b,e), (d,i)}
EXPANDE(h)	{c, g, h, j}	{a}	{j}	{a}	{(c,g), (h,j), (b,e), (d,i)}
EXPANDE(j)	{c, g, h, j}	{a}	{}	{a}	{(c,g), (h,j), (b,e), (d,i)}
TENTACONECTAR(a)	{c, g, h, j, a, b}	{}	{a,b}	{}	{(c,g), (h,j), (a,b), (d,i)}
EXPANDE(a)	{c, g, h, j, a, b, d}	{}	{b,d}	{}	{(c,g), (h,j), (a,b), (d,i)}
EXPANDE(b)	{c, g, h, j, a, b, d}	{e}	{d}	{e}	{(c,g), (h,j), (a,b), (d,i)}
EXPANDE(d)	{c, g, h, j, a, b, d, i}	{e}	{i}	{e}	{(c,g), (h,j), (a,b), (d,i)}
EXPANDE(i)	{c, g, h, j, a, b, d, i}	{e}	{}	{e}	{(c,g), (h,j), (a,b), (d,i)}
TENTACONECTAR(e)	{c, g, h, j, a, b, d, i}	{e}	{}	{}	{(c,g), (h,j), (a,b), (d,i)}

O seguinte teorema se aplica a grafos em geral, ponderados ou não. Entretanto, sua aplicação neste artigo é somente para árvores. Potencialmente, a ideia do algoritmo poderia ser utilizada para determinar o emparelhamento conexo máximo ponderado um grafo em geral.

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. □

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 um emparelhamento conexo máximo ponderado 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). A seguir, vamos expressar essas considerações em equações.

Considere v um vértice de T^r. Se v é folha em T^r, então $\overline{B_v} = B_v = 0$. Senão, B_v e $\overline{B_v}$ são definidos da seguinte forma.

$$\overline{B_v} = \sum_{u \in A(r,v)} B_u \tag{1}$$

$$B_v = max \left(\overline{B_a} + w(a,v) + \sum_{u \in A(r,v) \setminus \{a\}} B_u \right), \forall a \in A(r,v) \tag{2}$$

Note que, em (1), v está emparelhado com seu pai em T_v^r. Então, vamos considerar que todas as articulações u filhas de v estão emparelhadas em T_v^r. Por isso, temos o somatório de B_u. Já em (2), v está emparelhado com um de seus filhos. Logo, temos que encontrar o filho a que maximiza um emparelhamento máximo desse tipo em T_v^r. Se queremos representar um emparelhamento que possui a aresta (v, a), então temos que somar $\overline{B_a}$, já que a não está saturado em T_a^r. As outras articulações u filhas de v estão emparelhadas em T_u^r e, portanto, para cada uma, somamos B_u.

Mediante a essa formulação, a solução do problema consiste em calcular B_r, onde $r \in T$ é uma articulação qualquer.

O algoritmo descrito a seguir constrói, de maneira dinâmica, 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 no sentido das folhas para a raiz 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 no sentido de r para as 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.

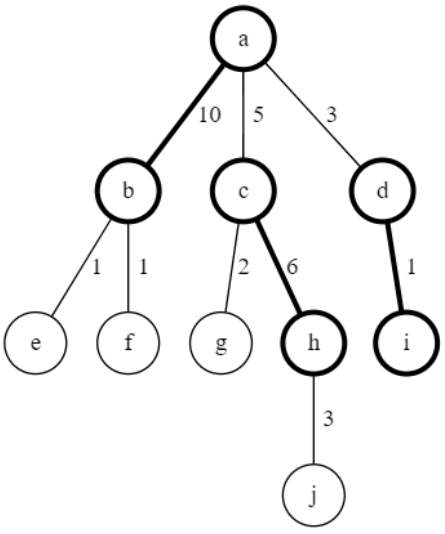


Figura 4: Um emparelhamento conexo máximo ponderado

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 4 ilustra uma árvore e seu emparelhamento conexo máximo ponderado de peso 17. Esse emparelhamento foi obtido com a execução do Algoritmo 2. A Tabela 2 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 3 e que a cardinalidade do emparelhamento ponderado é menor do que a do não ponderado.

Tabela 2: 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	–

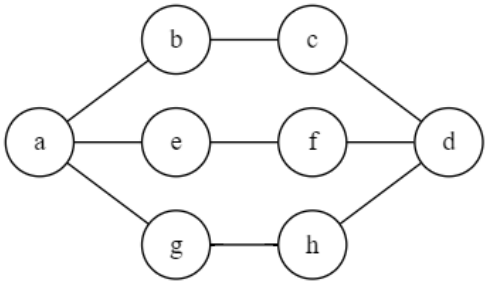


Figura 5: Um grafo com o número de separadores minimais exponencial em relação aos vértices

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 ao número de vértices. □

4. Um contraste entre emparelhamentos desconexos e conexos

A complexidade do problema de encontrar um emparelhamento desconexo máximo em um grafo qualquer G está em aberto. Uma idéia de solução é a de retirar do grafo, alternativamente, cada um dos seus separadores minimais compostos S, determinando o emparelhamento máximo M de $G - S$. O emparelhamento de maior cardinalidade dentre os obtidos é um emparelhamento desconexo máximo.

Entretanto, essa abordagem leva a um algoritmo que pode ser exponencial, pois o grafo pode ter um número exponencial de separadores minimais. Como exemplo desse tipo de grafo, observe que o grafo da Figura 5 possui separadores minimais de cardinalidade 3 contendo um elemento qualquer de cada um dos 3 conjuntos $\{b, c\}$, $\{e, f\}$ e $\{g, h\}$. Devido a essa dificuldade, conjecturamos que o problema seja NP-completo.

Apesar disso, para algumas classes, o problema pode ser resolvido em tempo polinomial como, por exemplo, ciclos, árvores e grafos bloco, para os quais foram encontrados algoritmos lineares em Masquio et al. [2019]. Nesse mesmo trabalho, foi mostrado um algoritmo de complexidade $O(nm)$ para grafos cordais. Foi mencionado ainda que não há solução para grafos completos ou split.

Em contraste, o problema de determinar um emparelhamento conexo máximo tem solução polinomial para grafos em geral. Neste trabalho mostramos ainda que, se for dado um emparelhamento máximo do grafo, o emparelhamento conexo máximo pode ser obtido com complexidade linear.

5. Conclusões

Neste artigo, discutimos os problemas de emparelhamento conexo máximo ponderado e não ponderado. Contribuimos com dois novos algoritmos lineares.

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

Evidenciamos que o problema de determinar um emparelhamento conexo máximo não ponderado 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 emparelhamento conexo máximo ponderado para grafos conexos, que permanece em aberto. A contribuição dada neste artigo é uma solução linear para o problema, quando restrito à classe de árvores.

Também mostramos as diferenças entre os problemas de emparelhamento máximo conexo e desconexo.

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

- Qual a complexidade do problema de emparelhamento conexo máximo ponderado para grafos conexos?
- Qual a complexidade do problema de emparelhamento desconexo máximo para grafos gerais?

Referências

Goddard, W., Hedetniemi, S. M., Hedetniemi, S. T., e Laskar, R. (2005). Generalized subgraph-restricted matchings in graphs. *Discrete Math.*, 293(1-3):129–138. ISSN 0012-365X.

Masquio, B., Pinto, P., e Szwarcfiter, J. (2019). Algoritmos eficientes para emparelhamentos desconexos em grafos cordais e grafos bloco. In *Anais do IV Encontro de Teoria da Computação*, Porto Alegre, RS, Brasil. SBC. URL https://sol.sbc.org.br/index.php/etc/article/view/6390.

Masquio, B. P. (2019). Emparelhamentos desconexos. Master's thesis, Universidade do Estado do Rio de Janeiro. Disponível em https://github.com/BMasquio/papers/raw/master/MastersThesisBrunoMasquio.pdf.

Micali, S. e Vazirani, V. V. (1980). An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *21st Ann. Symp. on Foundations of Comp. Sc.*, p. 17–27.

Savage, C. (1980). Maximum matchings and trees. *Information Processing Letters*, 10(4):202 – 205. ISSN 0020-0190. URL http://www.sciencedirect.com/science/article/pii/0020019080901404.