LI Simpósio Brasileiro de Pesquisa Operacional
Limeira–SP, 02 a 06 de setembro de 2019

EMPARELHAMENTOS DESCONEXOS EM GRAFOS CORDAIS E SUBCLASSES

Bruno Porto Masquio
Universidade do Estado do Rio de Janeiro, IME/UERJ
brunomasquio@gmail.com

Paulo Eustáquio Duarte Pinto[1]
Universidade do Estado do Rio de Janeiro, IME/UERJ
pauloedp@ime.uerj.br

Jayme Luiz Szwarcfiter[1]
Universidade Federal do Rio de Janeiro, PESC/COPPE e IM - UFRJ
Universidade do Estado do Rio de Janeiro, IME/UERJ
jayme@nce.ufrj.br

RESUMO

Os problemas de emparelhamentos máximos em grafos são bem estudados e trazem grandes contribuições à Teoria de Grafos, tanto do ponto de vista teórico quanto prático. Há numerosos estudos na área, destinados a emparelhamentos irrestritos e sem pesos. Mais recentemente, foram propostos os emparelhamentos restritos a subgrafos, que consideram propriedades dos subgrafos induzidos pelos vértices do emparelhamento. Neste trabalho, abordamos uma dessas novas propostas, que são os emparelhamentos desconexos, os quais procuram estudar emparelhamentos máximos tais que o subgrafo induzido pelos vértices do emparelhamento seja desconexo. Descrevemos algoritmos eficientes para resolver o problema em grafos cordais e subclasses, baseados em caracterizações teóricas. O algoritmo de grafos cordais tem complexidade de tempo $O(nm)$ e, além disso, foi possível uma redução para complexidade linear em algumas subclasses, tais como árvores e grafos bloco.

PALAVRAS CHAVE. Grafos Cordais, Algoritmos, Complexidade Computacional.

ABSTRACT

Graph matching problems are well studied and bring great contributions to Graph Theory from both the theoretical and practical points of view. There are numerous studies for unrestricted and weighted/unweighted matchings. More recently, subgraph-restricted matchings have been proposed, which consider properties of the subgraph induced by the matching vertices. In this paper, we approach one of these new proposals, disconnected matching, which seeks to study maximum matching, such that the subgraph induced by the matching vertices is disconnected. We have described efficient algorithms to solve the problem for chordal graphs and subclasses based on theoretical characterizations. The chordal graphs algorithm has $O(nm)$ time complexity and, moreover, we were able to reduce its complexity to linear for some subclasses, such as trees and block graphs.

KEYWORDS. Chordal Graphs, Algorithms, Computational Complexity

[1]Projeto parcialmente financiado por FAPERJ.

https://proceedings.science/p/106954?lang=pt-br

1. Introdução

Um emparelhamento em grafos é um conjunto de arestas não adjacentes duas a duas. Os problemas de emparelhamentos máximos em grafos são amplamente estudados há décadas pela comunidade científica e já possuem diversos resultados e aplicações importantes.

Esses estudos abordam o problema de diversas formas, como, por exemplo, para grafos ponderados ou não, para classes de grafos específicas e para emparelhamentos restritos a subgrafos. Essa última abordagem, recentemente proposta, procura resolver o problema tal que certas propriedades de subgrafos induzidos pelos vértices de um emparelhamento sejam válidas. Algumas das propriedades que esses subgrafos podem ter incluem que os mesmos sejam, por exemplo, 1-regular, acíclico, conexo ou desconexo. O problema do emparelhamento envolvendo essa última propriedade, ou seja, de que o grafo induzido pelos vértices do emparelhamento seja desconexo, estudado em [Masquio, 2019] e [Goddard et al., 2005], é o foco deste trabalho. Observe que é possível determinar emparelhamentos desconexos para ciclos, conforme o Teorema 1[Masquio, 2019].

Teorema 1. *Um ciclo C_n não tem emparelhamentos desconexos para $n < 6$ e, para $n \geq 6$, temos que a cardinalidade do emparelhamento desconexo máximo de C_n é $\lfloor (n-2)/2 \rfloor$.*

A determinação de emparelhamentos desconexos pode ajudar a resolver problemas que envolvem segurança e sigilo de informações. Por exemplo, suponha que uma empresa de produção de software deseja formar equipes de desenvolvedores nas quais é utilizada a prática de programação em par, originada do desenvolvimento ágil. Além disso, é desejado que as equipes sejam isoladas, de forma que participantes de equipes distintas não tenham tido contato anterior. Podemos formar um grafo a partir da relação de todas as pessoas que já tiveram contato entre si e, assim, determinar emparelhamentos desconexos para resolver o problema. Nesse caso, as equipes seriam representadas por componentes conexas do subgrafo e os pares de programadores, por arestas do emparelhamento.

Neste artigo, discutimos o problema para grafos em geral e apresentamos caracterizações e algoritmos relativos ao emparelhamento desconexo para grafos cordais e algumas subclasses.

2. Emparelhamentos desconexos para grafos em geral

Incialmente, vamos apresentar a notação utilizada. Considere G um grafo e M um emparelhameto de G. Denotamos $G[M]$ como o subgrafo induzido de G pelos vértices incidentes das arestas de M. Além disso, usamos $M(G)$ para representar um emparelhamento máximo em G e $\beta(G)$ para a sua cardinalidade. Analogamente, considere $M_d(G)$ um emparelhamento desconexo máximo e $\beta_d(G)$, a sua cardinalidade.

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 $G - S$. Se a e b são dois vértices que podem ser separados por S, então S é dito também como $a - b$ *separador*. 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. Da mesma forma, S é um $a - b$ *separador minimal* se S é um $a - b$ separador e não há subconjunto próprio de S que também seja um $a - b$ separador. Podemos não especificar o par de vértices do separador, referindo-nos a S simplesmente como um *separador de vértices minimal*. Observamos que todo separador minimal é um separador de vértices minimal para algum par de vértices, mas não o contrário. Além disso, considere *separador composto* como um separador de vértices cuja remoção deixa, pelo menos, duas componentes conexas não triviais. Como um caso específico, definimos uma *articulação composta* como um separador composto de cardinalidade unitária.

Observe que determinar um emparelhamento desconexo máximo de um grafo G pode ser simples quando o grafo possui uma das propriedades a seguir.

- Se G não possui separador composto, então G não possui emparelhamento desconexo.
- Se G é desconexo e contém mais de uma componente conexa não trivial, então $\beta(G) = \beta_d(G)$ e $M(G)$ também é $M_d(G)$.

Caso G possua uma única componente conexa não trivial C, então $\beta_d(G) = \beta_d(C)$ e $M_d(G)$ é $M_d(C)$. Por isso, limitaremos o estudo neste trabalho apenas a grafos conexos e não triviais.

Considerando r a menor cardinalidade de um separador composto de um grafo G, temos que $\beta_d(G) \geq \beta(G) - r$ [Goddard et al., 2005]. Mostraremos, a seguir, um método para determinar emparelhamentos desconexos máximos em um grafo geral.

Os separadores compostos são a chave para um algoritmo de obtenção de emparelhamentos desconexos pois, se M é um emparelhamento desconexo, os vértices M*-expostos* (isto é, aqueles que não fazem parte do emparelhamento) formam um separador composto, que contém um separador composto minimal. Assim, se os separadores compostos minimais do grafo puderem ser enumerados em tempo polinomial, então $\beta_d(G)$ pode ser determinado em tempo polinomial. Basta considerar cada um dos separadores compostos S e, por sua vez, determinar um emparelhamento máximo em $G-S$, o que pode ser feito em tempo polinomial. Os emparelhamentos correspondentes de maior cardinalidade são emparelhamentos desconexos máximos.

3. Emparelhamentos desconexos para grafos cordais

Nesta seção, apresentaremos resultados para a determinação de $\beta_d(G)$ e $M_d(G)$ para um grafo cordal qualquer G, de forma eficiente. Como dito anteriormente, só estamos tratando grafos cordais que têm separadores compostos e possuem uma única componente conexa não trivial.

Um grafo cordal pode ser representado, compactamente, por uma árvore especial, denominada *clique tree* [Tarjan e Yannakakis, 1984] [Lewis et al., 1989]. Conforme [Blair e Peyton, 1993] [Ibarra, 2009], definimos T como uma clique tree do grafo G se os vértices de T correspondem às cliques maximais de G e T possui a *propriedade de interseção de cliques*. Essa propriedade determina que, para quaisquer duas cliques maximais K e K' de G, o conjunto $K \cap K'$ está contido em todas as cliques maximais no caminho de K até K' em T. Também é provado em [Blair e Peyton, 1993] que as arestas (K, K') de T correspondem a todos os separadores de vértices minimais $K \cap K'$ de G. Portanto, para qualquer clique tree T de um grafo cordal G, os vértices e as arestas de T correspondem às cliques maximais e aos separadores de vértices minimais de G, respectivamente. Uma clique maximal de G corresponde a exatamente um vértice de T, enquanto um separador de vértices minimal de G corresponde a pelo menos uma aresta de T. Desse modo, como um grafo cordal G possui, no máximo, n cliques maximais [Fulkerson e Gross, 1965], então G possui, no máximo, $n-1$ separadores de vértices minimais.

Algoritmos para a geração de uma clique tree de um grafo cordal conexo podem ser feitos em tempo linear, conforme descrito em [Blair e Peyton, 1993], [BrandstÄdt et al., 1999] [Spinrad, 2003] e [Berry e Simonet, 2016].

Observe um grafo cordal G e uma clique tree T correspondente na Figura 1. As cliques maximais de G são vértices de T, como, por exemplo, o vértice b de T que corresponde à clique $\{2, 3, 5\}$ de G e o vértice c de T corresponde à clique $\{2, 4, 5, 7\}$ de G. Observe também que os separadores de vértices minimais de G correspondem às arestas de T, como por exemplo o separador $\{2, 5\}$ em G, que é a aresta (b, c) de T.

Sabemos que um grafo cordal G tem, no máximo, $n-1$ separadores de vértices minimais e, portanto, no máximo, essa mesma quantidade de separadores minimais. A partir do método para encontrar emparelhamentos desconexos máximos em grafos gerais, poderíamos remover cada

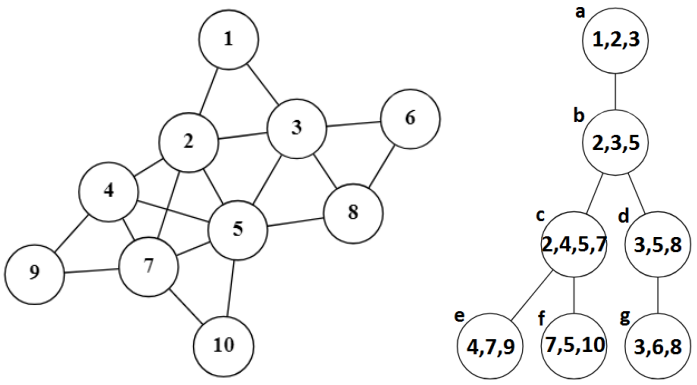


Figura 1: Um grafo cordal e uma clique tree correspondente

separador minimal S e calcular o emparelhamento para o grafo $G - S$. O separador minimal que maximiza $\beta(G - S)$ determina um emparelhamento desconexo máximo, que seria $M(G - S)$. Utilizando o algoritmo de [Micali e Vazirani, 1980] para calcular cada emparelhamento em $O(m\sqrt{n})$, a implementação completa para determinar um emparelhamento desconexo máximo teria complexidade de tempo $O(nm\sqrt{n})$.

Apresentaremos a seguir um algoritmo mais eficiente, com complexidade de tempo $O(nm)$, bem como sua base teórica e provas de corretude e de complexidade, estudados em [Masquio, 2019].

Para o desenvolvimento desse algoritmo, considere como base inicial necessária o Corolário 1, obtido com detalhes em [Masquio, 2019].

Corolário 1. *Seja G um grafo cordal e $\mathcal{S}$ o conjunto de separadores de vértices minimais de G. Então, $\beta(G - S) = \beta_d(G)$ e $M(G - S)$ é um emparelhamento desconexo máximo para algum $S \in \mathcal{S}$.*

Apresentaremos, a seguir, o Algoritmo 1, recursivo, que, a partir de uma clique tree T, relativa a um grafo cordal G, contendo, por definição, o conjunto de separadores de vértices minimais $\mathcal{S}$, identifica os emparelhamentos máximos de $G - S$ para todo $S \in \mathcal{S}$. Consequentemente, pelo Corolário 1, o algoritmo também identifica um emparelhamento desconexo máximo.

O algoritmo utiliza duas buscas em profundidade em T, identificando emparelhamentos de subgrafos induzidos em G.

No algoritmo, esses emparelhamentos serão armazenados nas variáveis globais $M_{a,b}$, que representam os emparelhamentos máximos no subgrafo induzido de G correspondente aos vértices representados na subárvore $T - (a, b)$ que contém a. Note que, ao retirar a aresta (a, b) de T, a clique tree será dividida em duas subárvores. Usaremos a notação A para representar a subárvore $T - (a, b)$ que contém o vértice a e B para representar a que contém o vértice b. Ao desconsiderar os vértices do separador (a, b), A e B representam, cada uma, pelo menos, uma componente conexa em G. Então, para determinar $M_{a,b}$, vamos denotar S_{AB} como o conjunto dos vértices de G que estão representados em A e não estão representados em B. Ou seja, basta considerarmos todos os vértices representados em A e retirarmos os vértices relativos ao separador (a, b), pois estes estão em B. Assim, calculamos o emparelhamento máximo do subgrafo induzido em G pelos vértices de S_{AB} e guardamos na variável $M_{a,b}$.

Como exemplo, considere o grafo cordal G e a clique tree T descritos na Figura 1. Vamos mostrar os dados que envolvem a determinação de $M_{c,b}$. A subárvore C é composta pelos vértices c, e, f e, B, por a, b, d, g. A aresta (c, b) de T representa o separador de vértices minimal $\{2, 5\}$ de G. O conjunto S_{CB} é igual a $\{4, 7, 9, 10\}$ e $M_{c,b}$ é um emparelhamento máximo do subgrafo induzido de S_{CB} em G. Nesse caso, só existe um emparelhamento máximo, $\{(4, 9), (7, 10)\}$.

Note que, se os dois subgrafos induzidos de um grafo cordal G pelos vértices de S_{AB} e de S_{BA} contêm, pelo menos, uma componente conexa não trivial cada, então (a, b) representa um separador composto. Além disso, podemos dizer que $M_{a,b} > 0$, $M_{b,a} > 0$ e, portanto, $M_{a,b} \cup M_{b,a}$ é um emparelhamento desconexo de G.

De acordo com o Corolário 1, $M(G - S)$ é um emparelhamento desconexo máximo para algum separador de vértices S, que está representado por alguma aresta (a, b) de T. Então, ao obtermos $M_{a,b}$ e $M_{b,a}$ para toda aresta (a, b) de T, conseguimos determinar um emparelhamento desconexo máximo. Basta encontrar uma aresta (a, b) que maximiza a soma $|M_{a,b}| + |M_{b,a}|$. Nesse caso, $M_d(G)$ seria $M_{a,b} \cup M_{b,a}$.

Para encontrar esses emparelhamentos do tipo $M_{a,b}$, o algoritmo usa duas buscas em profundidade a partir de um vértice v qualquer de T. A primeira busca processa as arestas no sentido das folhas à raiz de T^v (a árvore T enraizada em v) e a segunda, no sentido da raiz às folhas.

Para que não seja necessário calcular os emparelhamentos integralmente a partir do subgrafo induzido, usamos emparelhamentos parciais e os aumentamos de maneira eficiente. Nesse caso, se estamos processando a aresta (a, b), pela ordem de processamento das buscas descritas, para todos os vizinhos c de a, $c \neq b$, a aresta (a, c) de T já foi processada no sentido de c a a. Portanto, já possuimos o emparelhamento $M_{c,a}$. Sabemos que, ao retirar a aresta (a, b) de T, a árvore restante A possui todos os vértices de c representados, pois c está em A e a subárvore $A - (c, a)$ que contém c está contida em A.

Considere, então, $D_{a,b}$ como a diferença dos vértices de G que estão em a e não estão em b. Além disso, considere R como a união de todos os conjuntos de vértices S_{CA}, $c \neq b$, c vizinho de a. Note que, se considerarmos o subgrafo para gerar $M_{a,b}$, ou seja, o subgrafo induzido de G pelos vértices de S_{AB}, vamos ter exatamente $|D_{a,b}|$ vértices a mais que em R. Portanto, se considerarmos M a união de todos os $M_{c,a}$, podemos concluir que o emparelhamento $M_{a,b}$ poderá ser maior em, no máximo, $|D_{a,b}|$ arestas, ou seja, $|M_{a,b}| \leq |D_{a,b}| + |M|$. Então, para chegar eficientemente ao emparelhamento $M_{a,b}$, vamos usar M como um emparelhamento inicial e aumentá-lo até que seja máximo.

Vamos, então, emparelhar a clique $D_{a,b}$ em duas etapas. Na primeira etapa, começamos aumentando M de forma simples em $\lfloor |D_{a,b}|/2 \rfloor$ unidades ao emparelhar vértices dois a dois desse conjunto. Na segunda etapa, sabemos que M pode aumentar em, no máximo, $\lceil |D_{a,b}|/2 \rceil$ arestas até que seja máximo. Então, podemos usar um algoritmo que aumenta um emparelhamento em uma unidade com complexidade de tempo $O(m)$, descrito em [Micali e Vazirani, 1980], [Vazirani, 2012] e [Blum, 1990]. Esse algoritmo é baseado na detecção de caminhos M-aumentantes. Após as duas etapas, chegamos ao emparelhamento M máximo, que corresponde a $M_{a,b}$.

Para que calculemos os vértices usados nos subgrafos induzidos e os emparelhamentos parciais usados de forma eficiente, guardamos conjuntos de vértices acumulados e conjuntos de emparelhamentos acumulados da forma descrita a seguir. Para todos os conjuntos de vértices S_{BA} utilizados para gerar $M_{b,a}$, b vizinho de a, adicionamos S_{BA} à variável Vac_a e adicionamos todas arestas de $M_{b,a}$ a Mac_a.

A seguir, descreveremos mais detalhadamente as funções utilizadas no algoritmo.

- A função GERAEMPARELHAMENTODESCONEXO(T) é a função principal, que recebe uma

clique tree, faz as chamadas às funções BUSCA1 e BUSCA2, processa os resultados e, por fim, determina e retorna um emparelhamento desconexo máximo encontrado.

- A função EMPARELHACLIQUE(K) recebe um conjunto de vértices K cujo subgrafo induzido seja uma clique e os emparelha dois a dois. Ao final, esse emparelhamento é retornado.
- A função EMPARELHARESTO(H, M) recebe um grafo H e um emparelhamento parcial M, que pode ser máximo ou não em H. A função executa um loop que encontra caminhos M-aumentantes e aumenta M em uma unidade a cada iteração. Ao final do loop, M é máximo e é retornado. A determinação de um caminho M-aumentante pode ser implementada em $O(m)$ como descrito em [Micali e Vazirani, 1980], [Vazirani, 2012] e [Blum, 1990].
- A função BUSCA1(w, v) recebe dois vértices de uma clique tree T referentes a um grafo cordal e, de forma recursiva, visita o restante dos vértices. O processamento, feito no sentido das folhas à raiz, preenche as variáveis $M_{v,w}$ e, parcialmente, as variáveis Vac_w, Mac_w, para w pai de v na busca. A chamada inicial da função deve ser feita pela passagem do mesmo vértice nos dois parâmetros.
- A função BUSCA2(w, v) recebe dois vértices de uma clique tree referente a um grafo cordal e, de forma recursiva, visita o restante dos vértices. O processamento, feito no sentido da raiz às folhas, preenche as variáveis $M_{w,v}$, Vac_v, Mac_v, para w pai de v na busca. A chamda inicial da função deve ser feita pela passagem do mesmo vértice nos dois parâmetros.
- A função DIFERENCA(v, w) recebe dois vértices v e w de uma clique tree e retorna os vértices que estão representados em v que não estão representados em w.
- A função GRAFOINDUZIDO(G, S) recebe um grafo G e um conjunto de vértices S contidos em G e retorna o subgrafo induzido de S em G.

Algoritmo 1 Emparelhamento desconexo máximo em um grafo cordal

```
1:  função BUSCA2(Vértice: w, Vértice: u)
2:      se w ≠ u então
3:          D_{u,w} ← DIFERENCA(u, w)
4:          D_{w,u} ← DIFERENCA(w, u)
5:          H ← GRAFOINDUZIDO(G, Vac_w ∪ D_{w,u} − Vac_u − D_{u,w})
6:          M_{w,u} ← EMPARELHARESTO(H, EMPARELHACLIQUE(D_{w,u}) ∪ Mac_w − M_{u,w})
7:          Mac_u ← Mac_u ∪ M_{w,u}
8:          Vac_u ← Vac_u ∪ V(H)
9:      para a vizinho de u faça
10:         se a ≠ w então
11:             BUSCA2(u, a)
12:
13: função EMPARELHARESTO(Grafo: H, Emparelhamento: M)
14:     enquanto existir caminho M-aumentante em H faça
15:         P ← Caminho M-aumentante em H
16:         M ← M ⊕ P
17:     retornar M
18:
```

19: **função** EMPARELHACLIQUE(Conjunto de vértices: K)
20: $M_r \leftarrow \{\}$
21: **enquanto** existirem dois vértices u e w em K M_r-expostos **faça**
22: $M_r \leftarrow M_r \cup (u, w)$
23: **retornar** M_r
24:
25: **função** BUSCA1(Vértice: w, Vértice: u)
26: **para** a vizinho de u **faça**
27: **se** $a \neq w$ **então**
28: BUSCA1(u, a)
29: **se** $w \neq u$ **então**
30: $D_{u,w} \leftarrow$ DIFERENCA(u, w)
31: $H \leftarrow$ GRAFOINDUZIDO$(G, Vac_u \cup D_{u,w})$
32: $M_{v,w} \leftarrow$ EMPARELHARESTO$(H,$ EMPARELHACLIQUE$(D_{u,w}) \cup Mac_u)$
33: $Mac_w \leftarrow Mac_w \cup M_{u,w}$
34: $Vac_w \leftarrow Vac_w \cup V(H)$
35:
36: **função** GERAEMPARELHAMENTODESCONEXO(Clique Tree: T)
37: $v \leftarrow$ Vértice qualquer de T
38: BUSCA1(v, v)
39: BUSCA2(v, v)
40: $\beta_d(G) \leftarrow 0$
41: $M_d(G) \leftarrow \{\ \}$
42: **para** (u, w) aresta de T **faça**
43: **se** $|M_{u,w} \cup M_{w,u}| > \beta_d(G)$ e $|M_{u,w}| > 0$ e $|M_{w,u}| > 0$ **então**
44: $M_d(G) \leftarrow M_{u,w} \cup M_{w,u}$
45: $\beta_d(G) \leftarrow |M_d(G)|$
46: **retornar** $M_d(G)$
47:
48: #Dados: T: Clique Tree que representa um grafo cordal G
49: #Global:
50: #$M_{u,v}$: Emparelhamento, $(u, v) \in E(T)$
51: #Vac_v: Conjunto de vértices, $v \in V(T)$
52: #Mac_v: Emparelhamento, $v \in V(T)$
53: **para** v em $V(T)$ **faça**
54: $Vac_v \leftarrow \{\}$
55: $Mac_v \leftarrow \{\}$
56: $M \leftarrow$ GERAEMPARELHAMENTODESCONEXO(T)
57: $\beta_d(G) \leftarrow |M|$

A seguir, apresentaremos um exemplo de execução do algoritmo usando o grafo cordal G e a clique tree T descritos na Figura 1.

Inicialmente, são atribuídos conjuntos vazios a Mac_v e Vac_v, $v \in V(T)$. Executa-se GERAEMPARELHAMENTODESCONEXO(T), que retornará um emparelhamento desconexo.

Quando executamos GERAEMPARELHAMENTODESCONEXO(T), vamos atribuir a v o

vértice a de T, que será usado para a primeira chamada das funções BUSCA1(v, v) e BUSCA2(v, v).

Na primeira busca, são preenchidas, parcialmente, as variáveis de emparelhamentos acumulados Mac_u e de vértices acumulados Vac_u, $u \in V(T)$, além dos emparelhamentos $M_{u,w}$, u filho de w em T^v. Ao final dessa busca, teremos os valores das variáveis conforme a Tabela 1 e a Tabela 2.

Variável	Conteúdo	Variável	Conteúdo
Vac_a	$\{4, 5, 6, 7, 8, 9, 10\}$	Mac_a	$\{(4, 9), (7, 10), (6, 8)\}$
Vac_b	$\{4, 6, 7, 8, 9, 10\}$	Mac_b	$\{(4, 9), (7, 10), (6, 8)\}$
Vac_c	$\{9, 10\}$	Mac_c	$\{\}$
Vac_d	$\{6\}$	Mac_d	$\{\}$
Vac_e	$\{\}$	Mac_e	$\{\}$
Vac_f	$\{\}$	Mac_f	$\{\}$
Vac_g	$\{\}$	Mac_g	$\{\}$

Tabela 1: Tabela com o conteúdo das listas após a execução de BUSCA1(a, a)

Variável	Conteúdo	Variável	Conteúdo
$M_{a,b}$	$\{\}$	$M_{b,a}$	$\{(4, 9), (7, 10), (6, 8)\}$
$M_{b,c}$	$\{\}$	$M_{c,b}$	$\{(4, 9), (7, 10)\}$
$M_{b,d}$	$\{\}$	$M_{d,b}$	$\{(6, 8)\}$
$M_{c,e}$	$\{\}$	$M_{e,c}$	$\{\}$
$M_{c,f}$	$\{\}$	$M_{f,c}$	$\{\}$
$M_{d,g}$	$\{\}$	$M_{g,d}$	$\{\}$

Tabela 2: Tabela com o conteúdo das listas após a execução de BUSCA1(a, a)

Na segunda busca, são preenchidas as variáveis de emparelhamentos acumulados Mac_u e de vértices acumulados Vac_u, $u \in V(T)$, além dos emparelhamentos $M_{u,w}$, u pai de w em T^v. Ao final dessa execução, teremos os valores das variáveis conforme a Tabela 3 e a Tabela 4.

Variável	Conteúdo	Variável	Conteúdo
Vac_a	$\{4, 5, 6, 7, 8, 9, 10\}$	Mac_a	$\{(4, 9), (7, 10), (6, 8)\}$
Vac_b	$\{1, 4, 6, 7, 8, 9, 10\}$	Mac_b	$\{(4, 9), (7, 10), (6, 8)\}$
Vac_c	$\{1, 3, 6, 8, 9, 10\}$	Mac_c	$\{(1, 3), (6, 8)\}$
Vac_d	$\{1, 2, 4, 7, 6, 9, 10\}$	Mac_d	$\{(4, 9), (7, 10), (1, 2)\}$
Vac_e	$\{1, 2, 3, 5, 6, 8, 10\}$	Mac_e	$\{(1, 3), (6, 8), (2, 5)\}$
Vac_f	$\{1, 2, 3, 4, 6, 8, 9\}$	Mac_f	$\{(1, 3), (6, 8), (2, 4)\}$
Vac_g	$\{1, 2, 4, 5, 7, 9, 10\}$	Mac_g	$\{(4, 9), (7, 10), (1, 2)\}$

Tabela 3: Tabela com o conteúdo das listas após a execução de BUSCA2(a, a)

Variável	Conteúdo	Variável	Conteúdo
$M_{a,b}$	$\{\}$	$M_{b,a}$	$\{(4,9),(7,10),(6,8)\}$
$M_{b,c}$	$\{(1,3),(6,8)\}$	$M_{c,b}$	$\{(4,9),(7,10)\}$
$M_{b,d}$	$\{(4,9),(7,10),(1,2)\}$	$M_{d,b}$	$\{(6,8)\}$
$M_{c,e}$	$\{(1,3),(6,8),(2,5)\}$	$M_{e,c}$	$\{\}$
$M_{c,f}$	$\{(1,3),(6,8),(2,4)\}$	$M_{f,c}$	$\{\}$
$M_{d,g}$	$\{(4,9),(7,10),(1,2)\}$	$M_{g,d}$	$\{\}$

Tabela 4: Tabela com o conteúdo das listas após a execução de BUSCA2(a,a)

Ao final das buscas, temos preenchidos todos os emparelhamentos $M_{v,w}$ e $M_{w,v}$, v vizinho de w em T. Nesse momento, todas as arestas de T são analisadas e o emparelhamento desconexo máximo corresponderá a $M_{v,w} \cup M_{w,v}$ para os vértices v e w que maximizam a soma $|M_{v,w}| + |M_{w,v}|$ tais que $|M_{v,w}| > 0$ e $|M_{w,v}| > 0$.

No exemplo mostrado, temos duas arestas e, consequentemente, dois separadores de vértices que geram emparelhamento desconexo máximo em G. A primeira aresta, (b,c), representa o separador $\{2,5\}$, cuja remoção gera o emparelhamento desconexo máximo $\{(1,3),(6,8),(4,9),(7,10)\}$. A segunda, (b,d), representa o separador $\{3,5\}$, cuja remoção gera o emparelhamento desconexo máximo $\{(4,9),(7,10),(1,2),(6,8)\}$. A Figura 2 ilustra esses resultados.

A seguir vamos demonstrar a corretude do Algoritmo 1 e a sua complexidade de tempo.

Teorema 2. *O Algoritmo 1 identifica $M_d(G)$ e $M(G-S)$ para todo separador de vértices minimal S de um grafo cordal G.*

Demonstração. Para a prova, vamos considerar T a clique tree de entrada do algoritmo que representa o grafo cordal G.

Inicialmente, o algoritmo inicializa as variáveis Vac e Mac e chama a função GERAEMPARELHAMENTODESCONEXO(T), que retornará um emparelhamento desconexo máximo.

Um vértice v qualquer de T é escolhido para parâmetro da primeira chamada das buscas. A primeira busca determinará $M_{u,w}$ e a segunda, $M_{w,u}$, u filho de w em T^v.

Na função BUSCA1(w,u), u é filho de w, com exceção da primeira chamada em que $u = w$ e que não alterará as variáveis globais. Portanto, para determinar o emparelhamento $M_{u,w}$, geramos um grafo H com os vértices S_{UW} de G que estão representados em T^v_u (a subárvore enraizada em u de T^v), com exceção dos vértices do separador S que está representado pela aresta (u,w) de T.

Pela recursão, para cada vértice u visitado, u filho de w, adicionamos os vértices S_{UW} na variável Vac_w. Portanto, se estamos processando um vértice u filho de w e queremos calcular S_{UW}, basta considerarmos o conjunto da união de Vac_u com os vértices de u que não estão em w e que estão sendo guardados na variável $D_{u,w}$.

Além disso, para todo vértice u visitado, u filho de w, adicionamos o emparelhamento $M_{u,w}$ à variável Mac_w. Portanto, se estamos em um vértice u filho de w e queremos calcular o emparelhamento $M_{u,w}$, podemos considerar como Mac_u um emparelhamento inicial e tentar aumentá-lo até que ele seja máximo. Observe que aumentamos o grafo induzido, pelo acréscimo dos vértices de $D_{u,w}$.

Tanto na BUSCA1(w,u) quanto na BUSCA2(w,u), para aumentar um emparelhamento M em um grafo H até que tenhamos o emparelhamento máximo $M_{u,w}$, vamos usar duas técnicas.

https://proceedings.science/p/106954?lang=pt-br

Primeiro, vamos emparelhar todos os vértices que sabemos que estão expostos e que formam uma clique. Esse emparelhamento é feito em EMPARELHACLIQUE(K) e o adicionamos a M. Após isso, vamos aumentar achando caminhos M-aumentantes em H até que M seja máximo. Após essas duas etapas, temos $M_{u,w}$.

Na BUSCA2(w, u), temos um caso parecido com a busca anterior, porém devemos agora considerar que as variáveis Vac e Mac já estão parcialmente preenchidas. Nessa busca, u é filho de w, com exceção da primeira chamada em que $u = w$ e que não alterará as variáveis globais. Para determinar o emparelhamento $M_{w,u}$, geramos um grafo H com os vértices S_{WU} de G, que estão representados em $T^w - T^w_u$, com exceção dos vértices do separador S que está representado pela aresta (u, w) de T.

Assim como na primeira busca, se calculamos o emparelhamento $M_{w,u}$, adicionamos S_{WU} à variável Vac_u e o emparelhamento $M_{w,u}$ à variável Mac_u. A diferença está em, a partir das variáveis que temos, como determinar S_{WU} e o emparelhamento incial a ser aumentado.

Pela ordem de processamento dos vértices da segunda busca, se estamos calculando $M_{w,u}$, significa que, para todos os vizinhos a de u, $a \neq w$, Vac_u é exatamente a união de todos os S_{AU} e Mac_u é exatamente a união de todos os $M_{u,a}$. Além disso, Vac_w possui todos os vértices de $V(G)$, exceto os representados no próprio vértice w e Mac_w possui exatamente a união de todos os $M_{a,w}$, a vizinho de w.

Portanto, para obtermos S_{WU}, podemos considerar o conjunto Vac_w e retirar todos os vértices de T^w_u, ou seja, $Vac_u \cup D_{u,w}$. Também temos que adicionar os vértices de $D_{w,u}$, pois estes estão em w e não estão em u.

Para obtermos o emparelhamento M inicial, devemos considerar Mac_w e remover todas as arestas oriundas de T^w_u, ou seja, $M_{u,w}$. O aumento desse emparelhamento é feito da mesma maneira que na primeira busca.

Após a segunda busca, todas as variáveis do tipo $M_{u,w}$ estarão preenchidas e, então, podemos encontrar um emparelhamento desconexo máximo.

Usaremos o conceito que vimos no Corolário 1, considerando $\mathcal{S}$ o conjunto de separadores de vértices minimais de G. $M_d(G)$ pode ser obtido a partir de $M(G - S)$, para algum $S \in \mathcal{S}$.

Sabemos também que, segundo o que foi apresentado no início desta seção, cada aresta (a, b) de T representa um separador de vértices minimal em G correspondente à interseção dos vértices de G contidos em a e em b.

O loop da função GERAEMPARELHAMENTODESCONEXO(T) percorre todas as arestas (u, w) de T, buscando o emparelhamento desconexo de maior cardinalidade.

Seja (u, w) uma aresta de T que representa um separador de vértices minimal S de G. Vamos mostrar que só precisamos analisar (u, w) se $|M_{u,w}| > 0$ e $|M_{w,u}| > 0$. Discutimos anteriormente que, se $|M_{u,w}| > 0$ e $|M_{w,u}| > 0$, então $M_{u,w} \cup M_{w,u}$ é um emparelhamento desconexo. Por outro lado, se $|M_{u,w}| = 0$ ou $|M_{w,u}| = 0$, então, o emparelhamento $M_{u,w} \cup M_{w,u}$ pode ser desconexo ou não. Não é necessário considerar esse emparelhamento para encontrar um emparelhamento desconexo máximo no algoritmo, pois, se $M_{u,w} \cup M_{w,u}$ é um emparelhamento desconexo, há um separador minimal composto S_a contido nos vértices expostos de G. Além disso, $S_a \neq S$, pois S não é um separador composto. Por isso, mesmo que $M_{u,w} \cup M_{w,u}$ seja um emparelhamento desconexo máximo, há um emparelhamento de cardinalidade maior ou igual ao remover o separador S_a. Esse separador corresponde a alguma aresta (a, b) de T, que será analisada no algoritmo e $M_{a,b} \cup M_{b,a}$ será um possível resultado para o mesmo, pois $|M_{a,b}| > 0$ e $|M_{b,a}| > 0$.

Ao final, portanto, teremos encontrado $M_d(G)$, que será o emparelhamento de maior cardinalidade que satisfaça as condições enunciadas anteriormente. Além disso, encontramos $M(G -$

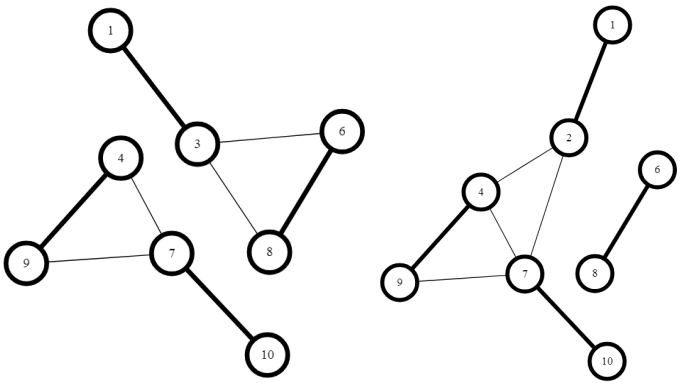


Figura 2: Emparelhamentos desconexos máximos do grafo da Figura 1

S) para todo separador de vértices minimal S. O emparelhamento $M(G - S)$ pode ser obtido a partir das variáveis encontradas, pois, sabendo que S é representado por alguma aresta (u, w) de T, então $M(G - S)$ é $M_{u,w} \cup M_{w,u}$. □

Teorema 3. *O Algoritmo 1 pode ser implementado em complexidade de tempo* $O(nm)$.

Maiores detalhes sobre o Teorema 3 podem ser obtidos em [Masquio, 2019].

4. Emparelhamentos desconexos para subclasses de grafos cordais

A classe de grafos bloco é uma subclasse de grafos cordais. Então, poderíamos calcular emparelhamentos desconexos máximos em $O(nm)$ usando o método para grafos cordais, descrito na Seção 3. A seguir, vamos apresentar uma idéia do método que leva a um algoritmo mais eficiente, com complexidade de tempo $O(m)$.

Sabemos que, em um grafo bloco B, u é um separador minimal se e somente se u é uma articulação. Portanto, $\beta(B) - 1 \leq \beta_d(B) \leq \beta(B)$. Definimos uma articulação composta livre v em B como uma articulação composta tal que $\beta(B - v) = \beta(B)$. Em outras palavras, se M é um emparelhamento máximo, v pode estar M-exposto. Nesse caso, podemos ver que $M(B - v)$ é um emparelhamento desconexo máximo.

Considere T uma clique tree que representa B. Podemos observar que os vértices de T representam as componentes biconexas de B e as arestas, as articulações de B. Também é possível ver que todo vértice de B está representado em uma estrela de T.

A partir de características estruturais de T, é possível determinar todas as articulações compostas livres de B em tempo linear usando duas buscas em profundidade de forma semelhante às feitas para grafos cordais. No caso de grafos bloco, não é necessário construir emparelhamentos baseados em caminhos alternantes, técnica que é usada iterativamente para grafos cordais. Se houver alguma articulação composta livre v, podemos usar um algoritmo de emparelhamentos máximos para grafos bloco em $B - v$, com complexidade $O(m)$. Portanto, é possível determinar um emparelhamento desconexo máximo em $O(m)$. Os algoritmos e as provas podem ser obtidos em [Masquio, 2019].

Para algumas outras subclasses de grafos cordais o problema é resolvido de forma simples, pois os separadores compostos minimais podem ser obtidos facilmente. Os resultados, obtidos em [Masquio, 2019], são dados pelos seguintes teoremas:

Teorema 4. *Um caminho P_n não tem emparelhamentos desconexos para $n < 5$ e, para $n \geq 5$ temos que $\beta_d(P_n) = \lfloor (n-1)/2 \rfloor$.*

Teorema 5. *Grafos completos e grafos split não têm emparelhamentos desconexos.*

5. Conclusões

Neste trabalho, consideramos emparelhamentos restritos a subgrafos e apresentamos algoritmos eficientes para encontrar emparelhamentos desconexos máximos em grafos cordais e subclasses.

O problema aqui tratado ainda é pouco estudado e permanece em aberto para grafos gerais. Desta forma, uma continuação natural seria ou desenvolver um algoritmo polinomial eficiente ou provar que esse problema pertence à classe NP-completo.

Referências

Berry, A. e Simonet, G. (2016). Computing a clique tree with algorithm MLS (maximal label search). *CoRR*, abs/1610.09623.

Blair, J. R. S. e Peyton, B. (1993). An introduction to chordal graphs and clique trees. In George, A., Gilbert, J. R., e Liu, J. W. H., editors, *Graph Theory and Sparse Matrix Computation*, p. 1–29, New York, NY. Springer New York. ISBN 978-1-4613-8369-7.

Blum, N. (1990). A new approach to maximum matching in general graphs. Technical report.

BrandstÄdt, A., Le, V., e Spinrad, J. (1999). *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics.

Fulkerson, D. R. e Gross, O. A. (1965). Incidence matrices and interval graphs. *Pacific J. Math.*, 15 (3):835–855.

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.

Ibarra, L. (2009). The clique-separator graph for chordal graphs. *Discrete Applied Mathematics*, 157(8):1737 – 1749. ISSN 0166-218X.

Lewis, J. G., Peyton, B. W., e Pothen, A. (1989). A fast algorithm for reordering sparse matrices for parallel factorization. *SIAM J. Sci. Stat. Comput.*, 10(6):1146–1173. ISSN 0196-5204.

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 $o(v|v|c|e|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, p. 17–27.

Spinrad, J. P. (2003). *Efficient graph representations*, volume 19. American Mathematical Society.

Tarjan, R. E. e Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13 (3):566–579. ISSN 0097-5397.

Vazirani, V. V. (2012). An improved definition of blossoms and a simpler proof of the MV matching algorithm. *CoRR*, abs/1210.4594.

https://proceedings.science/p/106954?lang=pt-br