

H-PARTIÇÕES DE GRAFOS

Arnaldo Cesar Almeida Oliveira

Universidade Federal Fluminense

Av. Gal. Milton Tavares de Souza, Instituto de Computação, 24210-346

arnaldoalmeida@id.uff.br, arnaldo.oliveira@cefet-rj.br

Fábio Protti

Universidade Federal Fluminense

Av. Gal. Milton Tavares de Souza, Instituto de Computação, 24210-346

fabio@ic.uff.br

Uéverton dos Santos Souza

Universidade Federal Fluminense

Av. Gal. Milton Tavares de Souza, Instituto de Computação, 24210-346

ueverton@ic.uff.br

RESUMO

Sejam G e H grafos simples e não direcionados. Este trabalho considera o problema de encontrar, se houver, uma partição $\mathcal{P} = \{S_1, S_2, \dots, S_k\}$ de $V(G)$ de modo que exista uma bijeção entre $\mathcal{P}$ e $V(H)$ satisfazendo a seguinte propriedade: se cada S_i é associado a $v_i \in V(H)$, então v_i é adjacente (resp., não adjacente) a v_j se e somente se S_i é completamente adjacente (resp., completamente não adjacente) a S_j . Se tal partição existe então dizemos que G possui uma H -partição. O foco deste trabalho é desenvolver métodos de reconhecimento de H -partições de um grafo e determinar algumas classes de grafos em que a solução para o Problema da H -Partição pode ser obtida em tempo polinomial.

PALAVRAS CHAVE. Partição em Grafos, H -partição, Complexidade de Algoritmos

Tópicos: Teoria dos Grafos, problemas de partição, classes de grafos

ABSTRACT

Let G and H be simple, undirected graphs. This work considers the problem of finding, if any, a partition $\mathcal{P} = \{S_1, S_2, \dots, S_k\}$ of $V(G)$ so that there is a bijection between $\mathcal{P}$ and $V(H)$ satisfying the following property: if each S_i is associated with $v_i \in V(H)$, then v_i is adjacent (resp., non-adjacent) to v_j if and only if S_i is completely adjacent (resp., completely non-adjacent) to S_j . If such a partition exists then G is said to have an H -partition. The focus of this work is to develop methods for recognizing H -partitions of a graph and to determine some classes of graphs for which the solution to the H -partition Problem can be obtained in polynomial time.

KEYWORDS. Graph Partitions, H -partition, Complexity of Algorithms

Paper topics: Graph Theory, partition problems, graph classes

1. Introdução

O estudo de problemas de partição de grafos é uma importante área da Teoria dos Grafos. A relevância do estudo dessa classe de problemas pode ser exemplificada por muitos motivos, dentre os quais pode-se destacar: (a) Dificuldade computacional: A maioria dos problemas de partição em grafos são NP-completos; portanto, a pesquisa nesses problemas pode levar ao desenvolvimento de algoritmos mais eficientes para casos particulares e à compreensão de sua complexidade computacional de forma mais refinada [Garey e Johnson, 1979; Papadimitriou, 1994]; (b) Aplicações práticas: Os problemas de partição em grafos têm várias aplicações, incluindo planejamento de redes, alocação de recursos, design de circuitos integrados, classificação de dados e processamento de imagens [Ribeiro et al., 2008; Fiduccia e Mattheyses, 1982]; (c) Redução de problemas: Os problemas de partição em grafos são úteis na construção de reduções de outros problemas a problemas de partição. Por exemplo, o problema de corte máximo em um grafo pode ser reduzido a um problema de particionamento de arestas [Karypis e Kumar, 1998; Leighton, 1983]; (d) Modelagem: Muitos problemas do mundo real podem ser formulados como problemas de partição em grafos, o que permite uma análise mais sistemática e precisa destes problemas [Van den Berg et al., 1997; Everitt, 1980]; (e) Desenvolvimento da teoria dos grafos: Os problemas de partição em grafos formam uma área de pesquisa ativa na Teoria dos Grafos, com várias aplicações em outras áreas da Matemática e da Ciência da Computação, como Álgebra, Geometria, Teoria da Computação e Análise de Algoritmos [Diestel, 2010; Lovász, 1979].

Além do exposto no parágrafo anterior, problemas de partição constituem-se em uma linha de pesquisa em constante evolução, com novos resultados sendo publicados regularmente e muitos problemas em aberto, garantindo que esse campo seja relevante e desafiador. Entre os problemas de partição mais conhecidos, podemos citar:

- Problema da k -partição de grafos (k -Partition): o objetivo é dividir o grafo em k conjuntos disjuntos de vértices, de tal forma que a soma dos pesos das arestas que ligam vértices de diferentes conjuntos seja maximizada ou minimizada [Garey e Johnson, 1979];

- Problema da M -partição (M -Partition Problem): consiste em determinar uma partição $\mathcal{P} = \{S_1, S_2, \dots, S_k\}$ do conjunto de vértices do grafo de forma que cada S_i satisfaça uma restrição interna de adjacências, e cada par $S_i, S_j, i \neq j$, satisfaça uma relação externa de adjacências. Estas relações estão codificadas em uma matriz $M_{k \times k}$ [Feder et al., 1999].

- Problema da partição mínima em cliques (*Minimum Clique Partition*): o objetivo é particionar o grafo em um número mínimo de cliques, de tal forma que cada vértice pertença a exatamente uma clique [Khuller e Raghavachari, 1994].

- Problema da partição mínima colorida (*Minimum Coloring Partition*): o objetivo é particionar o grafo em um número mínimo de conjuntos disjuntos de vértices, de tal forma que cada conjunto seja uma clique e os vértices de um mesmo conjunto tenham cores diferentes na coloração dada [Lin e Kernighan, 1973].

- Problema da H -Partição, introduzido por Dantas et al. [2005]: neste caso o intuito é particionar o conjunto de vértices de um grafo G em quatro subconjuntos A, B, C e D onde somente restrições externas entre as partes são relevantes, e tais restrições são consideradas por grafos-modelo H . As possíveis restrições consideradas entre as partes são: partes completamente adjacentes, partes completamente não adjacentes, ou sem qualquer restrição. Em Dantas et al.

[2005] é apresentado um algoritmo que resolve esse problema de modo eficiente para todas as possibilidades de H consideradas, exceto nos casos $H = 2K_2$ e $H = K_2 + S_2$.

Este trabalho trata do *Problema da H-Partição*, homônimo ao problema introduzido em [Dantas et al., 2005]; entretanto, o problema tratado nesse trabalho difere no fato de o grafo H poder ser de qualquer tipo, enquanto em Dantas et al. [2005] o grafo H deve sempre possuir quatro vértices. Além disso, no problema aqui considerado só são consideradas dois tipos de restrições externas entre as partes, quais sejam, ou as partes são completamente adjacentes ou completamente não adjacentes, o que não é o caso do problema homônimo em que outras restrições externas são tratadas.

O Problema da H -Partição tratado nesse trabalho, formalmente, consiste em: Dados dois grafos G e H , encontrar, se possível, um mapeamento $f : V(G) \rightarrow V(H)$ tal que se $f(u) \neq f(v)$, com $u, v \in V(G)$ então $uv \in E(G)$ se e somente se $f(u)f(v) \in E(H)$. Se esse mapeamento existe então $V(G)$ pode ser particionado em conjuntos não vazios $\{S_x : x \in V(H)\}$ tais que $S_x = \{u \in V(G) : f(u) = x\}$ e todos os vértices em S_x têm as mesmas relações de adjacência com qualquer vértice $w \notin S_x$ (assim, se $u, v \in S_x$ então $N(u) \setminus S_x = N(v) \setminus S_x$). Neste caso, diz-se que G admite uma H -partição $\mathcal{P} = \{S_x : x \in V(H)\}$, e define-se o problema correspondente da seguinte forma:

Problema da H-Partição

Entrada: Dois Grafos G e H .

Pergunta: G admite H -partição?

Em uma H -partição de G , cada S_x é um módulo de G . Assim, o Problema da H -Partição equivale a encontrar uma partição modular $\mathcal{P}$ de $V(G)$ tal que a contração de cada módulo de $\mathcal{P}$ em um único vértice produz o grafo H (H é o grafo quociente $G/\mathcal{P}$). A contração de um conjunto $S \subseteq V(G)$ é uma operação que resulta em um novo grafo G' no qual os vértices em S são substituídos por um novo vértice w tal que $N_{G'}(w) = N_G(S) \setminus S$.

O Problema da H -Partição também pode ser visto como uma variação do Problema da M -Partição, introduzido por Feder et al. [1999, 2003], definido como se segue. Seja M uma matriz $k \times k$ na qual todo elemento pertence ao conjunto $\{0, 1, *\}$. Diz-se que o grafo G admite uma M -partição se $V(G)$ pode ser coberto por conjuntos mutuamente disjuntos $S_1, \dots, S_k$ (alguns deles possivelmente vazios) tais que:

- (i) se $m_{ii} = 0$ (respectivamente, $1, *$) então S_i é um conjunto independente (respectivamente, uma clique, um conjunto sem restrições de adjacência interna);
- (ii) se $m_{ij} = 0$ para $i \neq j$ (respectivamente, $1, *$) então S_i e S_j são completamente não adjacentes (respectivamente, completamente adjacentes, sem restrições de adjacência entre si).

Muitos problemas combinatórios importantes podem ser expressos como casos particulares do Problema da M -Partição. Por exemplo, o Problema da k -Coloração é expresso por uma matriz com as entradas diagonais iguais a 0 e as entradas fora da diagonal iguais a $*$; O Problema do Reconhecimento de Grafos *Split*, por uma matriz 2×2 com as entradas diagonais sendo 0 e 1, e entradas fora da diagonal iguais a $*$. O Problema da H -Partição, no caso, é equivalente ao caso do

Problema da M -Partição no qual todos os conjuntos são não vazios e M é obtida pela substituição dos elementos diagonais da matriz de adjacência do grafo por $*'s$.

O problema de Isomorfismo de Grafos reduz-se trivialmente ao Problema da H -Partição. Dada uma entrada (G, H) para Isomorfismo de Grafos com $|V(G)| = |V(H)|$, é claro que G é isomorfo a H se e somente se G admite uma H -partição. Assim, o Problema da H -Partição é Isomorfismo-Difícil (*Graph Isomorphism Hard*).

O Problema da H -Partição, portanto, mostra-se interessante por residir entre os problemas de particionamento de grafos e ainda não ter a sua complexidade totalmente resolvida no caso geral. Este trabalho ater-se-á a apresentar um algoritmo que resolve o problema da H -Partição em tempo exponencial, e apresentará alguns casos em que o problema pode ser resolvido em tempo polinomial.

2. Preliminares

Alguns conceitos são fundamentais para desenvolver o tema deste trabalho. Nesta seção serão apresentadas as definições necessárias. Uma fonte de referência para o estudo de decomposições modulares é [Habib e Paul, 2010].

Um *módulo* de um grafo G é um conjunto $S \subseteq V(G)$ tal que para qualquer $x \notin S$, ou $S \subseteq N(x)$ ou $S \cap N(x) = \emptyset$. Um módulo S é *trivial* se $S = V(G)$ ou $S = \{u\}$ para um vértice $u \in V(G)$. Um módulo S é *forte* se S não faz *overlap* com nenhum outro módulo do grafo, i.e., para qualquer módulo S' de G vale que ou $S \cap S' = \emptyset$ ou $S \subseteq S'$ ou $S' \subseteq S$. Um módulo $S \subset V(G)$ é *maximal* se não existe módulo S' tal que $S \subset S' \subset V(G)$. Uma partição $\mathcal{P} = \{S_1, \dots, S_k\}$ de $V(G)$, para $k \geq 2$, é uma *partição modular maximal* de G se todo S_i é um módulo forte maximal de G . Pode-se mostrar que tal partição é única. Além disso, se todo S_i consiste de um único vértice, G é chamado de *grafo primo*. O menor grafo primo é o grafo P_4 .

Dada uma partição modular maximal $\mathcal{P} = \{S_1, \dots, S_k\}$ de $V(G)$, o grafo obtido pela contração de cada S_i a um único vértice v_i (de forma que v_i e v_j são adjacentes (resp., não adjacentes) se e somente se S_i e S_j são completamente adjacentes (resp., completamente não adjacentes)) é chamado de *grafo quociente* de G por $\mathcal{P}$, e denotado por G' ou $G|\mathcal{P}$.

A partir deste ponto, será conveniente considerar partições ordenadas. Para denotar uma partição ordenada de $V(G)$, escreve-se $(S_1, \dots, S_k)$. Por exemplo, se $k = 2$ então (S_1, S_2) e (S_2, S_1) são partições distintas.

Seja (G, H) uma entrada do Problema da H -Partição, e seja $\mathcal{M} = (M_1, \dots, M_k)$ uma partição modular maximal (ordenada) de H . Todas as definições que se seguem são relacionadas a esta partição.

Uma *ancoragem* é uma tupla $A = (u_1, \dots, u_k)$ dos vértices de G tais que u_i e u_j são adjacentes se e somente se os módulos M_i e M_j são adjacentes em $H|\mathcal{M}$. Esta definição implica que $G[\{u_1, \dots, u_k\}]$ é trivialmente isomorfo a $H|\mathcal{M}$.

Seja $A = (u_1, \dots, u_k)$ uma ancoragem. Para um índice $i \in \{1, \dots, k\}$, diz-se que um vértice $v \in V(G)$ é *similar* a u_i se $A' = (u_1, \dots, u_{i-1}, v, u_{i+1}, \dots, u_k)$ é também uma ancoragem.

Seja $\mathcal{A} = (A_1, \dots, A_k)$ uma partição ordenada de $V(G)$. Diz-se que $\mathcal{A}$ é uma *partição admissível* de G se toda tupla $(u_1, \dots, u_k)$ satisfazendo $u_i \in A_i, i \in \{1, \dots, k\}$, é uma ancoragem. Isto implica que, para toda ancoragem $(u_1, \dots, u_k)$ com $u_i \in A_i$, vale que A_i é formado por vértices que são similares a $u_i, i \in \{1, \dots, k\}$.

Exemplo 1. Seja $T = (u_1, \dots, u_k)$ uma ancoragem. Então, várias partições admissíveis da forma $(A_1, \dots, A_k)$ podem ser construídas com $u_i \in A_i$ para todo i . Por exemplo, suponha que H contém $k = 3$ componentes conexas e G contém $\ell = 4$ componentes conexas $G_1, \dots, G_4$. Escolha $u_i \in S_i = V(G_i)$, $1 \leq i \leq 4$, e considere a tupla $T = (u_1, \dots, u_3)$. Então T é uma ancoragem e $(S_1 \cup S_4, S_2, S_3)$, $(S_1, S_2 \cup S_4, S_3)$, e $(S_1, S_2, S_3 \cup S_4)$ são partições admissíveis. $\square$

3. Um primeiro algoritmo

Nesta seção será apresentado um algoritmo (Algoritmo 1) que resolve o problema da H -Partição. O algoritmo é exponencial, mas para certos grafos H é executado em tempo polinomial.

Uma abordagem para resolver o Problema da H -Partição consiste em determinar uma partição modular maximal ordenada $\mathcal{M} = (M_1, \dots, M_k)$ de $V(H)$, e tentar construir uma partição modular $\mathcal{P} = (S_1, \dots, S_\ell)$ de $V(G)$ testando ancoragens $T = (w_1, \dots, w_k)$ de vértices de G como se segue: para cada ancoragem, considere todas as possíveis (se existirem) partições admissíveis $\mathcal{A} = (A_1, \dots, A_k)$ de $V(G)$ tais que A_i contém w_i e vértices que são similares a w_i , $i \in \{1, \dots, k\}$. Se $\mathcal{A} = (A_1, \dots, A_k)$ é admissível então o processo deve continuar resolvendo o Problema da H -partição com entradas $(G[A_i], H[M_i])$, $1 \leq i \leq k$. Este é basicamente o modo de funcionamento do Algoritmo 1, apresentado na próxima página. Se H é desconexo ou $\overline{H}$ é desconexo, o processo de buscar partições admissíveis pode ser simplificado. Para ilustrar, considere o seguinte exemplo.

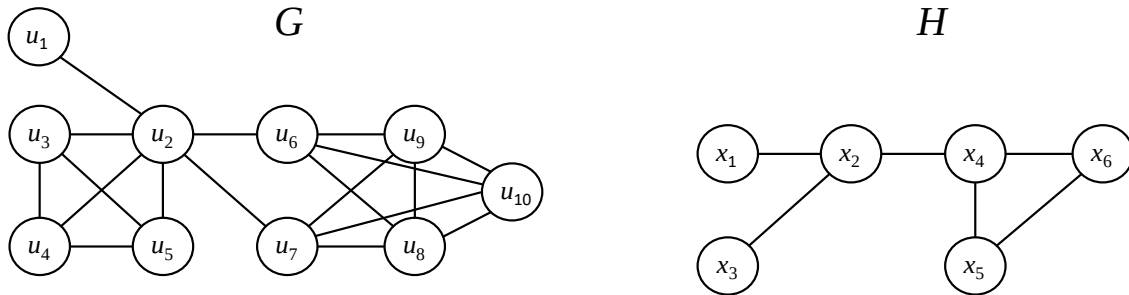


Figure 1: Exemplo de entrada para o Algoritmo 1.

Na Fig. 1, uma partição modular maximal ordenada de H é $\mathcal{M} = (M_1, M_2, M_3, M_4)$ tal que $M_1 = \{x_1, x_3\}$, $M_2 = \{x_2\}$, $M_3 = \{x_4\}$ e $M_4 = \{x_5, x_6\}$. Considere a tupla $T = (u_1, u_2, u_6, u_{10})$ de vértices de G . Então, T é uma ancoragem e $\mathcal{A} = (A_1, A_2, A_3, A_4)$, com $A_1 = \{u_1, u_3, u_4, u_5\}$, $A_2 = \{u_2\}$, $A_3 = \{u_6, u_7\}$ e $A_4 = \{u_8, u_9, u_{10}\}$, é uma partição admissível. Repete-se então o processo tomando como entradas $(G[A_i], H[M_i])$, $1 \leq i \leq 4$.

Teorema 2. O Algoritmo 1 está correto.

Prova. O caso base do algoritmo é $V(H) = 1$; neste caso, o algoritmo trivialmente retorna $V(G)$. Na linha 3, a partição modular maximal de H é computada. O algoritmo agora verifica todas as partições admissíveis de G . As linhas 5-8 correspondem ao caso de H ser desconexo. O algoritmo determina partições admissíveis como ilustrado no Exemplo 1, fazendo combinações de k elementos dentre as componentes conexas de G . As linhas 9-12 correspondem a um caso análogo ($\overline{H}$ é desconexo). As linhas 13-18 testam todas as possíveis tuplas de k vértices de G , seleciona as que

Algoritmo 1: PARTIÇÃO(G, H)

Entrada: Grafos G, H com $|V(G)| \geq |V(H)| \geq 1$

Saída: Uma H -partição $\mathcal{P}$ de G , ou $\mathcal{P} = \emptyset$ se G não admite H -partição

```

1 se  $|V(H)| = 1$  então
2   retorna  $\mathcal{P} = V(G)$ 

3 Computar uma partição modular maximal ordenada  $\mathcal{M} = (M_1, \dots, M_k)$  de  $H$ 
4  $\mathcal{A} \leftarrow \emptyset$ 
5 se  $H$  é desconexo então
6   Sejam  $G_1, \dots, G_\ell$  as componentes conexas de  $G$ 
7   se  $\ell \geq k$  então
8      $\mathcal{A} \leftarrow$  todas partições ordenadas possíveis  $(A_1, \dots, A_k)$  of  $V(G)$  onde
       $A_i = \cup_{j \in I_i} V(G_j), I_i \subseteq \{1, \dots, \ell\}$ 
9 se  $\overline{H}$  é desconexo então
10  Sejam  $G_1, \dots, G_\ell$  componentes conexas de  $\overline{G}$ 
11  se  $\ell \geq k$  então
12     $\mathcal{A} \leftarrow$  todas partições ordenadas possíveis  $(A_1, \dots, A_k)$  de  $V(G)$  onde
       $A_i = \cup_{j \in I_i} V(G_j), I_i \subseteq \{1, \dots, \ell\}$ 
13 se ambos  $H$  e  $\overline{H}$  são conexos então
14   para cada tupla  $(u_1, \dots, u_k)$  de vértices de  $V(G)$  faça
15     se  $(u_1, \dots, u_k)$  é uma ancoragem então
16        $A_i \leftarrow$  vértices similares a  $u_i, 1 \leq i \leq k$ 
17       se  $(A_1, \dots, A_k)$  é uma partição admissível então
18         inclua  $(A_1, \dots, A_k)$  in  $\mathcal{A}$ 
19 para cada  $(A_1, \dots, A_k) \in \mathcal{A}$  faça
20   computar  $G_i = G[A_i]$  e  $H_i = H[M_i], 1 \leq i \leq k$ 
21    $\mathcal{P}_i \leftarrow$  PARTIÇÃO( $G_i, H_i$ ),  $1 \leq i \leq k$ 
22   se  $\mathcal{P}_i \neq \emptyset$  para cada  $1 \leq i \leq k$  então
23     retorna  $\mathcal{P} = \mathcal{P}_1 \cup \dots \cup \mathcal{P}_k$ 
24 retorna  $\mathcal{P} = \emptyset$ 

```

são ancoragens, e constrói uma partição admissível para cada ancoragem. Todas as partições admissíveis ficam armazenadas em $\mathcal{A}$, em qualquer dos três casos acima. A seguir, o algoritmo testa recursivamente se G_i admite uma H_i -partição, para $1 \leq i \leq k$, onde G_i e H_i são os subgrafos induzidos pelos vértices de A_i e M_i , respectivamente. Se G admite uma H -partição, então deve haver no mínimo uma partição admissível $\mathcal{A}$ para o qual todos os testes na linha 22 retornam resultado diferente de vazio; neste caso, o algoritmo retorna $\mathcal{P} = \mathcal{P}_1 \cup \dots \cup \mathcal{P}_k$. Por outro lado, caso G não admita uma H -partição, na linha 22 alguma chamada recursiva retornará $\mathcal{P}_i = \emptyset$. $\square$

4. Casos em que o Problema da H -partição pode ser resolvido em tempo polinomial

Para algumas entradas G e H o problema da H -partição pode ser resolvido em tempo polinomial. Alguns casos são bem simples, mas serão tratados nesta seção pois podem servir de apoio para casos mais complexos.

Proposição 3. *Se $|V(H)| \leq c$ para uma constante positiva c , então o Problema da H -Partição pode ser resolvido em tempo polinomial.*

Prova. Verificando o Algoritmo 1, o número de partições admissíveis a serem testadas nesse caso é polinomial, pois $k \leq c$. Além disso, a profundidade da recursão é limitada por uma constante. Logo, a proposição segue. $\square$

Necessitaremos da seguinte definição. Dado um conjunto $S \subseteq V(G)$, $\overline{N}(S)$ é o conjunto dos vértices que têm algum não-vizinho em S .

Teorema 4. *O Problema da H -Partição pode ser resolvido em tempo polinomial se $H = C_3$.*

Prova. Inicialize $A := a$, $B := \emptyset$ e $C := \emptyset$, onde a é um vértice de G escolhido arbitrariamente. Se $\overline{N}(A) \neq \emptyset$, adicione os vértices de $\overline{N}(A)$ a A , e repetidamente aplique a mesma operação enquanto a condição $\overline{N}(A) \neq \emptyset$ for verdadeira. No final do processo, se $V(G) \setminus A = \emptyset$ então G não admite H -partição. Caso contrário, todo par de vértices $x \in A, y \in V(G) \setminus A$ satisfaz $xy \in E(G)$. Escolha então um vértice arbitrário $b \in V(G) \setminus A$ e faça $B := b$. Se $\overline{N}(B) \neq \emptyset$, adicione vértices de $\overline{N}(B)$ a B , e repita o processo enquanto a condição $\overline{N}(B) \neq \emptyset$ for verdadeira. No final do processo, se $V(G) \setminus (A \cup B) = \emptyset$ então G não admite H -Partição. Caso contrário, todo par de vértices $x \in (A \cup B), y \in V(G) \setminus (A \cup B)$ satisfaz $xy \in E(G)$. Nesta última situação, escolher $C := V(G) \setminus (A \cup B)$. O processo todo é executado em tempo polinomial. $\square$

Proposição 5. *Seja H um grafo sem arestas tal que $|V(H)| = k$. Então, G admite H -partição se e somente se G possui $p \geq k$ componentes conexas.*

Prova. Se G admite H -partição então é claro que G possui $p \geq k$ componentes conexas. Por outro lado, sejam $S_1, S_2, \dots, S_p$ os subconjuntos de vértices associados às componentes conexas de G e $V(H) = \{w_1, \dots, w_k\}$, com $k \leq p$. Então pode-se determinar um mapeamento, sem perda de generalidade, da seguinte forma: $f(S_1) = w_1, \dots, f(S_{k-1}) = w_{k-1}, f(S_k) = f(S_{k+1}) = \dots = f(S_p) = w_k$. Portanto, G admite H -partição. $\square$

Proposição 6. *G admite uma H -partição se e somente se $\overline{G}$ admite uma $\overline{H}$ -partição.*

Prova. Decorre da definição de H -Partição. $\square$

Uma prova alternativa do Teorema 4 consiste em usar as Proposições 5 e 6. Verificar se G admite uma C_3 -partição é equivalente a verificar se $\overline{G}$ admite uma I_3 -partição, onde I_3 é o grafo sem arestas com três vértices. Isso ocorre se $\overline{G}$ possui pelo menos três componentes conexas.

Teorema 7. *O Problema da H -partição pode ser resolvido em tempo polinomial se $H = C_4$.*

Prova. Decidir se G admite uma C_4 -partição, pela Proposição 6, equivale a decidir se $\overline{G}$ admite uma $2K_2$ -partição. Como o grafo $2K_2$ tem duas componentes conexas, $\overline{G}$ deve ser uma união disjunta de dois grafos G_1 e G_2 tais que cada G_i admite uma K_2 -partição. Pela Proposição 6 novamente, $\overline{G_i}$ deve admitir uma I_2 -partição, $i = 1, 2$. Finalmente, pela Proposição 5, $\overline{G_i}$ deve possuir pelo menos duas componentes conexas, $i = 1, 2$. Todos estes testes podem ser realizados em tempo polinomial. $\square$

Teorema 8. *Se H é um grafo primo então G admite uma H -partição se e somente se G' é isomorfo a H .*

Prova. Suponha que G admita uma H -partição, e que $V(H) = \{w_1, \dots, w_k\}$. Seja $\mathcal{P} = \{S_1, S_2, \dots, S_k\}$ uma partição modular de G tal que $f(S_i) = w_i$, $1 \leq i \leq k$. O mapeamento f nos diz que o grafo quociente de G pela partição modular $\mathcal{P}$ é H . Pela primalidade de H , temos que todos os pares (x, y) com $x \in S_i$ e $y \in S_j$, $i \neq j$, têm vizinhanças diferentes em $V(G) \setminus (S_i \cup S_j)$. Logo, cada S_i é módulo maximal e forte, isto é, $\mathcal{P}$ é a partição modular de G . Portanto, o grafo quociente de G por $\mathcal{P}$ é G' , e, além disso, G' é isomorfo a H . Reciprocamente, se G' é isomorfo a H , os conjuntos da partição modular maximal de G fornecem de forma imediata o modo de construir o mapeamento de $V(G)$ para $V(H)$, de acordo com a definição de H -partição. Logo G admite H -partição. $\square$

Corolário 9. *O Problema da H -partição pode ser resolvido em tempo polinomial se H é um caminho P_k , $k \geq 4$, ou um ciclo C_l , $l \geq 5$.*

Prova. Como os grafos do enunciado são primos, a prova segue trivialmente do Teorema 8. $\square$

Vamos agora mostrar que se H é uma árvore, então o Problema da H -partição pode ser resolvido em tempo polinomial. Para a demonstração deste resultado, alguns conceitos são necessários. É importante observar que o isomorfismo entre árvores é um problema que pode ser resolvido em tempo polinomial usando, por exemplo, o algoritmo de busca em profundidade [Cormen et al., 2009]. Entretanto, este trabalho usará como base a técnica de código de isomorfismo que pode ser encontrada em Valiente [2002]. Para tal, é necessário definir o que é o código de isomorfismo de uma árvore.

Definição 10. Seja T uma árvore não ordenada com n nós. O código de isomorfismo de T é a sequência de n inteiros dada por $\text{cod}[T] = [\text{size}[T]; \text{cod}[w_1]; \dots, \text{cod}[w_k]]$, onde os nós $w_1, \dots, w_k$ são os filhos da raiz de T da direita para a esquerda, arranjados em ordem lexicográfica crescente do código de isomorfismo. O código de isomorfismo de uma árvore é o código de isomorfismo da raiz dessa árvore. Um exemplo pode ser visto na Figura 2.

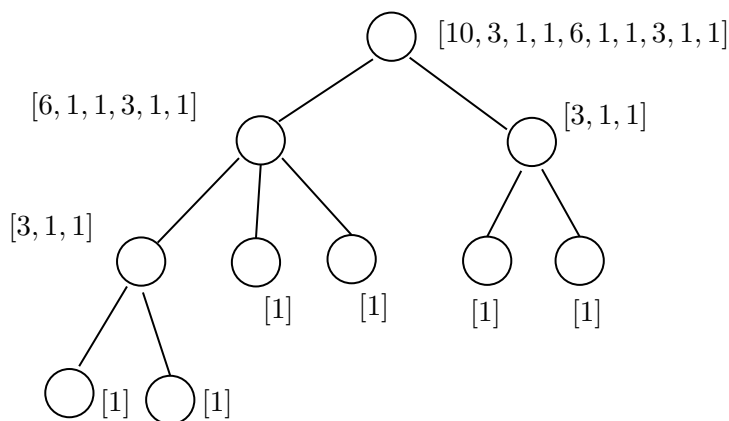


Figure 2: Uma árvore e seu código de árvore.

Precisamos agora analisar as partições modulares maximais de árvores. Dada uma árvore T e sua partição modular maximal $\mathcal{P}$, o grafo quociente de T por $\mathcal{P}$ é também uma árvore, denotada por T' e que chamaremos de *árvore quociente* de T . Note que T' não possui conjuntos de folhas irmãs (folhas com o mesmo vizinho) e, se tiver quatro ou mais nós, é um grafo primo. Será útil considerar uma versão ligeiramente modificada do código de árvore de T' :

Definição 11. O **código modificado** da árvore quociente T' de uma árvore T define-se da seguinte forma. Observe que folhas irmãs de T serão contraídas em um único vértice de T' . O código modificado de uma folha w_i de T' será denotado por $\text{cod}[w_i] = 1(k)$, onde k é o número de elementos do conjunto de folhas irmãs que contém w_i na árvore original T . As demais regras de construção do código modificado seguem a Definição 10. Veja um exemplo na Figura 3.

Antes de demonstrar o Teorema 13, precisamos do seguinte resultado:

Proposição 12. Se G admite uma H -partição, então G admite uma H' -partição.

Prova. Seja f o mapeamento de $V(G)$ em $V(H)$, de acordo com a definição de H -partição. Suponha $f(S_i) = v_i$, $v_i \in V(H)$. Considere agora os módulos M_j da partição modular maximal de H , e suponha que M_j está associado a $u_j \in V(H')$. Seja f' o mapeamento de $V(G)$ em $V(H')$ definido por $f'(\cup_{v_i \in M_j} S_i) = u_j$, $u_j \in V(H')$. Este mapeamento nos diz que G admite uma H' -partição. $\square$

Teorema 13. Se H é uma árvore então o Problema da H -partição pode ser resolvido em tempo polinomial.

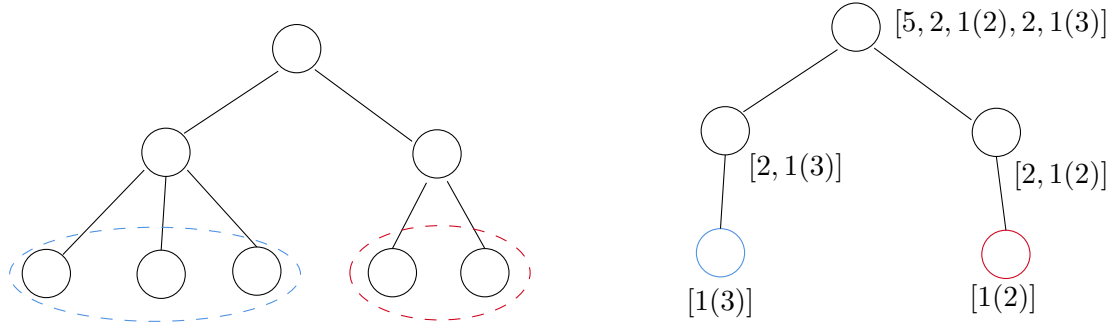


Figure 3: Árvore T à esquerda e sua árvore quociente T' à direita, com o código modificado.

Prova. Podemos supor que o grafo H não é trivial. Pela Proposição 12, se G admitir uma H -partição, então G admite uma H' -partição. Como H é uma árvore, H' é também uma árvore. Analisamos dois casos.

(a) Se $|V(H')| \leq 3$, verificamos facilmente se G admite H' -partição usando técnicas já descritas acima. Se G não admite H' -partição, então G não admite H -partição. Caso contrário, H' tem duas folhas, e verificamos os subconjuntos S_1 e S_2 de $V(G)$ que foram mapeados para as folhas de $V(H')$. Considere o código modificado de H' , e suponha que os códigos modificados de suas folhas sejam $1(k)$ e $1(l)$. Então é fácil ver que se algum S_i induz um grafo com pelo menos k componentes conexas, e S_j ($j \neq i$) induz um grafo com pelo menos l componentes conexas, G admite H -partição. Senão, G não admite uma H -partição.

(b) Se $|V(H')| \geq 4$, H' é uma árvore que é um grafo primo. Se G admitir H -partição, pelo Teorema 8, G' deve ser isomorfo a H' . Verificamos então se G' é uma árvore e, a seguir, testamos então o isomorfismo das árvores G' e H' . Podemos realizar este teste utilizando os seus códigos de isomorfismo [Valiente, 2002]. Se o teste não for satisfeito, G não admite H -partição. Caso contrário, verificamos a bijeção entre folhas de G' e H' , construída pelo teste de isomorfismo, e consideramos os códigos modificados destas folhas. Para toda folha v de G' mapeada a uma folha $f(v)$ de H' , com códigos modificados $1(k)$ e $1(l)$, respectivamente, deve valer $k \geq l$. Assim, G admite H -partição se e somente se esta condição é satisfeita. Todos os testes nos casos (a) e (b) podem ser realizados facilmente em tempo polinomial. $\square$

Exemplo 14. Para ilustrar o caso (b) do teorema anterior, considere $\text{cod}[G'] = [5, 2, 1(2), 2, 1(3)]$ e $\text{cod}[H'] = [5, 2, 1(2), 2, 1(2)]$. Então, G admite H -partição.

5. Conclusão

Neste trabalho, foi iniciado o estudo do Problema da H -Partição. Foi apresentado um algoritmo que resolve o problema em tempo exponencial para o caso geral, e alguns casos em que esse problema pode ser resolvido em tempo polinomial. Observa-se que o problema da H -Partição tem forte relação com decomposição modular e isomorfismo de grafos. Uma questão importante é conseguir mais informações sobre a complexidade do problema; até o momento, sabemos apenas que é Isomorfismo-Difícil. Futuramente, pretende-se estudar outros casos importantes, como H sendo um cografo (ou um grafo de distância hereditária).

References

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. (2009). *Introduction to Algorithms*. MIT Press.
- Dantas, S., Figueiredo, C., Gravier, S., e Klein, S. (2005). Finding h-partitions efficiently. *ITA*, 39: 133–144.
- Diestel, R. (2010). *Graph Theory*. Springer-Verlag.
- Everitt, B. (1980). *Cluster Analysis*. Wiley.
- Feder, T., Hell, P., Klein, P., e Motwani, R. (1999). Complexity of graph partition problems. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing - STOC'99*, p. 464–472, New York. Plenum Press.
- Feder, T., Hell, P., Klein, P., e Motwani, R. (2003). List partitions. *SIAM Journal on Discrete Mathematics*, 16(3):449–478.
- Fiduccia, C. M. e Mattheyses, R. M. (1982). A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, p. 175–181.
- Garey, M. R. e Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.
- Habib, M. e Paul, C. (2010). A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59. URL <https://doi.org/10.1016/j.cosrev.2010.01.001>.
- Karypis, G. e Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392.
- Khuller, S. e Raghavachari, B. (1994). Finding disjoint paths in expander graphs and applications. *SIAM Journal on Computing*, 23(6):1276–1289.
- Leighton, F. T. (1983). *Complexity Issues in VLSI*. MIT Press.
- Lin, S.-t. e Kernighan, B. W. (1973). Graph coloring problems. *Journal of the ACM (JACM)*, 20(2):261–264.
- Lovász, L. (1979). *Combinatorial Problems and Exercises*. North-Holland.
- Papadimitriou, C. H. (1994). *Computational Complexity*. Addison-Wesley.
- Ribeiro, C. C., Uchoa, E., e Werneck, R. F. (2008). The prize-collecting steiner forest problem. *Discrete Applied Mathematics*, 156(18):3602–3612.
- Valiente, G. (2002). *Tree Isomorphism*, p. 151–251. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-662-04921-1. URL https://doi.org/10.1007/978-3-662-04921-1_4.

Van den Berg, J., Kreveld, M. V., Overmars, M., e Schwarzkopf, O. (1997). *Computational Geometry: Algorithms and Applications*. Springer-Verlag.