Particularidades do Problema de Partição na Convexidade P_3 em Grafos com *Treewidth* Limitada

Lorrana F. de Castro[1], **Rodolfo A. de Oliveira** [1], **Fábio Protti**[2]

[1]Instituto do Noroeste Fluminense / Universidade Federal Fluminense – (INF/UFF)
Av. João Jasbick, S/N – Aeroporto – S. A. Pádua/RJ – Brasil.

[2]Instituto de Computação / Universidade Federal Fluminense – (IC/UFF)
Gal. Milton Tavares de Souza, S/N – S. Domingos – Niterói/RJ – Brasil.

{lorranafilemes,rodolfooliveira}@id.uff.br,fabio@ic.uff.br

Abstract. *In this work, we present an algorithm that solves partitioning problems on P_3 convexity for graphs with limited treewidth. Given a graph $G = (V, E)$ and $X \subseteq V$, we say that X is a p_3-convex set if any vertex with at least two neighbors in X is also an element of it. The collection of all p_3-convex sets contained in V constitutes the P_3 convexity of G. The p-partition problem on P_3 convexity consists of partitioning V into p non-empty and pairwise disjoint p_3-convex sets, for an integer $p \geq 2$. Despite the* NP*-completeness even for split graphs, we show that such a partitioning problem is solvable in linear time when the treewidth of a graph and p are both limited.*

Resumo. *Neste trabalho, apresentamos um algoritmo que resolve o problema de partição na convexidade P_3 quando restrito a grafos com treewidth limitada. Dados um grafo $G = (V, E)$ e $X \subseteq V$, dizemos que X é um conjunto p_3-convexo se todo vértice com pelo menos dois vizinhos em X também pertence a X. A coleção de todos os subconjuntos p_3-convexos de V constitui a convexidade P_3 em G. O problema da p-partição na convexidade P_3 consiste em particionar V em p conjuntos p_3-convexos não-vazios e mutualmente disjuntos, para um inteiro $p \geq 2$. Apesar da* NP*-completude mesmo para grafos split, mostramos que o problema é resolvido em tempo linear para grafos com treewidth e o número p limitados.*

1. Introdução

Da ideia de contágio por infecção descrita em [Centeno et al. 2009] surgiu o conceito de intervalo que compõe a convexidade P_3. Basicamente, a ideia consistia de uma região contendo quadras infectadas, onde uma quadra não infectada se tornaria infectada caso fosse adjacente a pelo menos duas que fossem. Assim, fica clara a regra de disseminação da enfermidade. Assim, dado um grafo com conjunto de vértices V, um conjunto $X \subseteq V$ é chamado p_3-convexo se qualquer vértice de V com pelo menos dois vizinhos em X também pertence a X. Além disso, já em [Centeno et al. 2010], há questionamentos sobre partições na convexidade P_3 onde, para um dado inteiro positivo p, somos desafiados a encontrar uma divisão para V em p conjuntos p_3-convexos. Por trás desse conceito está o interesse de identificar regiões endêmicas, visto que uma infecção numa tal região não atingiria outras. Todavia, este problema foi demonstrado NP-completo, mesmo restrito a grafos *split*. Por esta razão, atacamos o problema considerando aspectos de complexidade

tratável por parâmetro fixo (FPT - "*fixed-parameter tractable*") em grafos com *treewidth* limitada.

2. Algumas Formalizações

Considere $G = (V, E)$ um grafo finito, onde V é o conjunto de vértices e E o conjunto de arestas. Denote por $|V| = n$ e $|E| = m$. Para $X \subseteq V$, o subgrafo induzido por X é denotado por $G[X]$. Dado um vértice $u \in V$, defina a **vizinhança de** u por $N(u) = \{v \in V : uv \in E\}$, e a **vizinhança fechada de** u por $N[u] = N(u) \cup \{u\}$. Para $X \subseteq V$, denote $N[X] = \bigcup_{\forall u \in X} N[u]$. O **grau de** u em G é definido como $d(u) = |N(u)|$. Um conjunto $X \subseteq V$ é dito uma **cobertura de vértices** quando toda aresta de G incide em algum vértice de X. Defina $cv(G)$ como a **cardinalidade da menor cobertura de vértices de** G. Um subconjunto de vértices X de V é definido como p_3**-convexo** se para qualquer vértice $v \in V \setminus X$ temos $|N(v) \cap X| \leq 1$. A **convexidade** P_3 relativa a V é a coleção de todos os subconjuntos de V que sejam p_3-convexos. Dado $X \subseteq V$, o **fecho** p_3**-convexo** de X é o menor conjunto p_3-convexo que contém X. O processo de encontrar o fecho p_3-convexo para um subconjunto de vértices X pode ser efetuado de forma iterativa, executando o seguinte processo: enquanto houver $v \in V \setminus X$ de modo que $|N(v) \cap X| > 1$, faça $X \leftarrow X \cup \{v\}$. Dado um inteiro positivo $p \geq 2$, uma p**-partição na convexidade** P_3 é uma coleção de conjuntos p_3-convexos $\{X_1, X_2, \ldots, X_p\}$ tal que: (i) $X_i \neq \emptyset, \forall 1 \leq i \leq p$; (ii) $X_i \cap X_j = \emptyset, \forall 1 \leq i \neq j \leq p$; e (iii) $\bigcup_{i=1}^{p} X_i = V$.

Uma fórmula em **Lógica Monádica de Segunda Ordem** (LMSO) é uma fórmula que admite quantificadores ($\forall$ ou $\exists$) em relações unárias, usualmente na relação de pertinência ou inclusão. As relações não quantificadas podem ser variáveis de testes lógicos sobre alguma propriedade relacionada ao problema ao qual se refere a fórmula, sendo imprescindível que sejam verificáveis em tempo $O(1)$ para o tipo de algoritmo de objetivamos. Portanto, alguns problemas em grafos podem ser reescritos como fórmulas em LMSO. Por exemplo, $X \subseteq V : \forall x, y \in V(\neg adj(x, y) \vee x \in X \vee y \in X)$ é uma fórmula em LMSO que verifica se X é uma cobertura de vértices, onde $adj(x, y)$ é um teste lógico que verifica se x e y são adjacentes em G, o que pode ser verificável em $O(1)$ caso a matriz de adjacência seja conhecida.

A fórmula do exemplo anterior é chamada de LMSO de **classificação 1** (LMSO_1) por haver apenas quantificação sobre o conjunto de vértices do grafo. Quando habilitamos também a quantificação sobre conjunto de arestas chamamos de **classificação 2** (LMSO_2). Muitos o problemas em grafos podem ser escritos em LMSO_2 mas não em LMSO_1, como por exemplo o problema do ciclo hamiltoniano. Agora vejamos a seguinte definição:

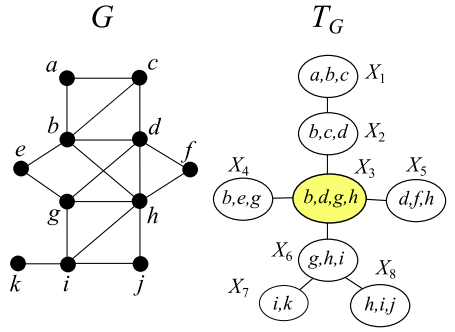
Definição 2.1 *Uma* ***árvore de decomposição*** *de um grafo $G = (V, E)$ é um par $(\mathcal{X}, \mathcal{T})$, onde $\mathcal{X} = \{X_1, \ldots, X_r\}$ com $X_i \subseteq V$ e $\mathcal{T}$ é uma árvore cujos vértices estão representados em $\mathcal{X}$ de modo que: (i) $\bigcup_{i=1}^{r} X_i = V$, (ii) se $uv \in E$, então $u, v \in X_i$, para algum $1 \leq i \leq r$ (i.e., toda aresta possui seus extremos em algum membro de $\mathcal{X}$) e (iii) se $v \in X_i$ e $v \in X_j$, então $v \in X_{j'}$ para todos j' no caminho de X_i a X_j em $\mathcal{T}$ (i.e., os membros de $\mathcal{X}$ contendo v induzem uma subárvore de $\mathcal{T}$).*

Cada conjunto X_i é chamado de ***bag***. A **largura** de uma árvore de decomposição é o tamanho de sua maior *bag* menos uma unidade, e a ***treewidth*** de um grafo é a menor largura dentre todas as árvores de decomposições do mesmo. Defina $tw(G)$ como o valor da *treewidth* de G. Para dado grafo G, denotaremos por T_G uma árvore de decomposição

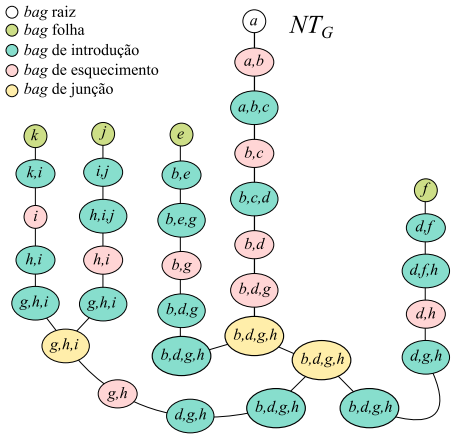
para G. Note que o número de arestas de G é $O(kn)$, onde $tw(G) = k$. Na Figura 1(a), temos um exemplo de um grafo G e uma árvore T_G.

Uma propriedade interessante conhecida como Teorema de Courcelle é que:

Teorema 2.1 *[Courcelle 1990] Assuma que φ seja uma fórmula em LMSO$_2$ e G um grafo com n vértices. Além disso, suponha que uma árvore T_G com largura k também seja fornecida. Então existe um algoritmo que verifica se φ satisfaz G em tempo $f(||\varphi||, k).n$, para alguma função f.*



(a) Um grafo G com $tw(G) = 3$ e uma árvore T_G, onde X_3 é a maior *bag* e $|X_3| = 4$.



(b) Uma árvore de decomposição legal NT_G.

Figura 1. Exemplo de árvore de decomposição e árvore de decomposição legal.

Uma outra definição importante sobre árvores de decomposição é:

Definição 2.2 *Uma* **árvore de decomposição legal** *(onde, aqui, "legal" tem o sentido do termo original "nice" da literatura) é uma árvore de decomposição $(\mathcal{X}, \mathcal{T})$ onde $\mathcal{X} = \{X_1, \ldots, X_r\}$, $\mathcal{T}$ é uma árvore enraizada e: (i) toda bag de $\mathcal{T}$ tem no máximo dois filhos, (ii) se uma bag X_i tem dois filhos $X_j, X_{j'}$, então $X_i = X_j = X_{j'}$, (iii) se uma bag X_i tem uma filho X_j, então ou $|X_i| = |X_j| + 1$ e $X_j \subseteq X_i$ (dizemos X_i introduz v, se $X_i \setminus X_j = \{v\}$), ou $|X_i| = |X_j| - 1$ e $X_j \subseteq X_i$ (dizemos X_i esquece v, se $X_j \setminus X_i = \{v\}$).*

Para simplificar as nomenclaturas das *bags*, denominamos *bag* **de introdução** (**esquecimento**) toda *bag* que introduz (esquece) algum vértice na árvore de decomposição $(X, \mathcal{T})$ no sentido *bottom-up*. Ademais, denominamos *bag* **de junção** toda *bag* com dois filhos e *bag* **folha** toda *bag* sem filho. Para G, denote NT_G uma árvore de decomposição legal para G. Assim, para o mesmo grafo da Figura 1(a), apresentamos NT_G na Figura 1(b).

Proposição 2.1 *[Cygan et al. 2015] Se G admite uma árvore de decomposição com largura no máximo k, então G também admite uma árvore de decomposição legal de largura no máximo k. Ademais, dada T_G com largura máxima k, podemos encontrar NT_G em tempo $O(k^2 \max\{n, |\mathcal{X}|\})$ cujo número de bags é $O(kn)$.*

3. Resultados

Primeiramente, mostramos a conversão do problema da p-partição na convexidade P_3 em LMSO$_2$. Para melhor entendimento, dividimos a fórmula em partes F_i:

F_1**:** p conjuntos como variáveis livres: $X_1, X_2, \ldots, X_p \subseteq V$;

F_2**:** os conjuntos são não-vazios: $\bigwedge_{i=1}^{p} \exists x \in X_i$;

F_3**:** qualquer vértice de V está em alguma partição e os conjuntos são mutuamente disjuntos: $\forall x \in V \left(\bigvee_{i=1}^{p} (x \in X_i) \wedge \bigwedge_{i \neq j} (\neg x \in X_i \vee \neg x \in X_j) \right)$

F_4**:** os conjuntos são p_3-convexos:

$$\forall x \in V \left(\bigwedge_{i \neq j} (x \in X_i \Rightarrow \neg (y, z \in X_j \Rightarrow adj(x, y) \wedge adj(x, z)) \right)$$

Assim, a expressão fica:

$$F_1 : F_2 \wedge F_3 \wedge F_4$$

Assim, podemos garantir pelo Teorema de Courcelle que existe um algoritmo que resolve o problema de p-partição na convexidade P_3 em tempo $f(||\varphi||, k).n$, para uma função f. Note que, entretanto, f também está em função de p, pois o comprimento de φ depende de p. Vale ressaltar que o teorema nos garante que existe um algoritmo, mas não nos exibe claramente qual é este algoritmo. Assim, desenvolvemos um algoritmo $\mathcal{A}$ produzindo os seguintes resultados:

Teorema 3.1 *O Algoritmo $\mathcal{A}$ resolve o problema da p-partição na convexidade P_3 em grafos com $tw(G) = k$ com no máximo $O(2^{2k+3} \cdot p^{k+2p+2}) \cdot n$ passos.*

Corolário 3.1 *O Algoritmo $\mathcal{A}$ resolve em tempo linear o problema da p-partição na convexidade P_3 em grafos com $tw(G) = k$ e p ambos fixados.*

4. Conclusão

O Algoritmo $\mathcal{A}$ traz resultados interessantes para o problema quando aplicado a grafos *split*. Como mencionado, o problema da p-partição na convexidade P_3 é NP-completo para tal classe de grafos. Contudo, quando p é fixado, temos que existem pelo menos $p-1$ vértices no conjunto independente de grau máximo 1 ou a clique do grafo não possui mais do que p vértices, implicando sua *treewidth* no máximo $p-1$. Logo, podemos resolver em tempo linear o problema em grafos *split* para p fixado.

Outra classe interessante em que o Algoritmo $\mathcal{A}$ tem boa performance é aquela formada por grafos cuja cobertura de vértices é limitada por k, pois $tw(G) \leq cv(G)$ [Fiala et al. 2011]. Logo $\mathcal{A}$ resolve o problema da p-partição na convexidade P_3 quando o tamanho da cobertura de vértices do grafo é limitado.

Referências

Centeno, C. C., Dantas, S., Dourado, M. C., Rautenbach, D., e Szwarcfiter, J. L. (2010). Convex Partitions of Graphs induced by Paths of Order Three. *Discrete Mathematics and Theoretical Computer Science*, 12(5):175–184.

Centeno, C. C., Dourado, M. C., e Szwarcfiter, J. L. (2009). On the convexity of paths of length two in undirected graphs. *Electronic Notes in Discrete Mathematics*, 32:11 – 18. DIMAP Workshop on Algorithmic Graph Theory.

Courcelle, B. (1990). The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75.

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., e Saurabh, S. (2015). *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition.

Fiala, J., Golovach, P. A., e Kratochvíl, J. (2011). Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theoretical Computer Science*, 412(23):2513 – 2523.