Universidade Federal do Rio de Janeiro - PESC/COPPE
Professora: Celina M. Herrera de Figueiredo
Disciplina: CPS845 - Tópicos Especiais em Teoria dos Grafos
Aluna: Mariana Martins Ferreira da Cruz

Nota de aula - Recognition of quasi-Meyniel graphs - Aula 08

1 Resumo:

Este texto tem por objetivo descrever as notas da aula intitulada como *Recognition of quasi-Meyniel graphs*, oitava aula referente ao curso Tópicos Especiais em Teoria dos Grafos (2020), oferecido pelo programa de em Engenharia de Sistemas e Computação PESC/COPPE - UFRJ e ministrado pela professora Celina de Figueiredo.

1.1 Problemas propostos:

Além de documentar o conteúdo abordado ao longo da aula, este documento se propõe a resolver as seguinte questões:

1. Considere o grafo da Figura 16 ($\overline{C_6}$) o texto. O grafo contém algum cap? (p.4) O grafo é um grafo Meyniel?
2. Qual é a complexidade do algoritmo para testar a existência de um cap? (p.2)
3. Faça o passo a passo do algoritmo para testar a existência de um cap, usando como entrada o grafo da Figura 16 do texto. (p.2)

2 Grafos quasi-Meyniel

Definição 1. *Define-se por* ***buraco [hole]*** *um ciclo sem corda com pelo menos quatro vértices.*

Definição 2. *Define-se por* ***cap*** *um ciclo com pelo menos* 5 *vértices que possui apenas uma corda, que é triangular.*

Dizemos que um grafo G contém um grafo H, quando H é um subgrafo induzido de G. Um grafo é livre de H se não contém H como subgrafo.

Definição 3. *Um grafo é dito* ***grafo de Meyniel*** *se todo ciclo ímpar com ao menos* 5 *vértices possui pelo menos duas cordas.*

Definição 4. *Seja G um grafo. Um vértice $x \in V(G)$ é dito* ***tip*** *se x é tal que a corda de cada cap em G tem x como um vértice extremo.*

Informalmente, podemos dizer que um tip é um vértice que supervisiona as cordas triangulares de ciclos ímpares.

Definição 5. *Um grafo G é dito* ***grafo quasi-Meyniel*** *se não contém nenhum buraco ímpar e contém um tip.*

È interessante notar que ao removermos o vértice tip de um grafo quasi-Meyniel o grafo resultante é um grafo de Meyniel.

Fica claro a partir das definições que todo grafo Meyniel é também um grafo quasi-Meyniel, onde cada um de seus vértices é por vacuidade um tip, já que não há ciclos com apenas uma corda.

Por outro lado, todo grafo quasi-Meyniel que não é Meyniel deve conter um ciclo impar com corda triangular, um cap. Além disso, deve conter no máximo dois tips.

Se G é um grafo quase-Meyniel com um vértice tip x, então todo subgrafo induzido de G que contém x é também quase-Meyniel com tip x e todo subgrafo induzido de G que não contém x é Meyniel.

2.1 Algoritmo de Reconhecimento

O algoritmo utilizado para reconhecimento da Grafos quasi-Meyniel tem complexidade polinomial. Seu procedimento é baseado em duas estratégias:

1. encontrar um vértice tip de um gráfico quasi-Meyniel;
2. Fazer uma decomposição cortes de vértices a partir desse vértice tip para reconhecer o grafo quasi-Meyniel.

   Este procedimento está bem detalhado em [3].

   Observe que, ao encontrar as cordas de todas os caps de um grafo G, pode-se facilmente encontrar um candidato a tip de G. No entanto, isso é insuficiente para reconhecer grafos quase-Meyniel, uma vez que ainda é necessário verificar se há um buraco ímpar que passa pelo vértice tip, e este problema é NP-completo em geral. [1].

2.1.1 Encontrando um cap

Para encontrar um cap em um grafo G (ou verificar se G não contém nenhum cap) pode-se utilizar a seguinte estratégia:

Para cada aresta $xy \in E(G)$ e para cada vértice $z \in N(x) \cap N(y)$, verificamos se x e y ambos têm um vizinho na mesma componente C de $G \setminus ((N(x) \cap N(y)) \cup N(z))$.

Se essa condição for satisfeita, então o conjunto de vértices $\{x; y; z\} \cup V(P)$, onde P é um caminho mais curto em C cujo um vértice de extremidade é adjacente a x e o outro a y, induz um cap com a corda xy.

Se a condição não se verifica para todas as opções de nós entre x; y e z, então G não contém um cap.

A priori o algoritmo encontra um cap qualquer, podendo ser obtido a partir de um ciclo impar ou de um ciclo par.

Note que a complexidade deste processo algorítmico para encontrarmos caps em um grafo G com n vértices e m arestas tem complexidade $O(n^3m)$, pois a análise das vizinhanças dos extremos de todas as arestas é feita em tempo $O(n^2m)$, e a análise para verificar que, para cada vértice $z \in N(x) \cap N(y)$ tanto x quanto y possuem vizinhos na componente C de G é feita em em tempo linear no tamanho do conjunto de vértices.

A seguir, exibiremos o passo a passo do algoritmo acima dando como entrada o grafo da Figura 1.

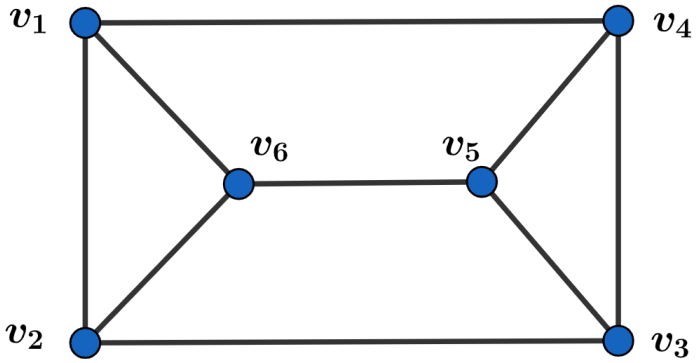

Figura 1: Grafo G, Figura 16 em [2].
Ilustração feita no software Geogebra.

Primeiramente, analisaremos a aresta v_1v_2 e a vizinhança de seus extremos:

$$N(v_1) = \{v_2, v_4, v_6\}, N(v_2) = \{v_1, v_3, v_6\}.$$

Note que v_6 é um vértice comum às vizinhanças de ambos os vértices. Assim, $v_6 \in N(v_1) \cap N(v_2)$. Note que $N(v_6) = \{v_1, v_2, v_5\}$.

O próximo passo do algoritmo é analisar se v_1 e v_2 possuem vizinhos em comum na componente $C = G \setminus ((N(v_1) \cap N(v_2)) \cup N(v_6))$. Dessa forma, temos

$$G \setminus \{v_6\} \cup \{v_1, v_2, v_5\} = \{v_3, v_4\}.$$

Note que, em $C = \{v_3, v_4\}$ tanto o vértice v_1 quanto o vértice v_2 possuem vizinhos. Desta forma, observe que o conjunto de vértices $\{v_1, v_2, v_6\} \cup \{v_3, v_4\}$ (destacado em vermelho), onde $v(P) = \{v_3, v_4\}$ (o caminho mais curto em C, visto que C é isomorfo a P_2) induz um cap com a corda v_1v_2, destacada em verde na Figura 2.

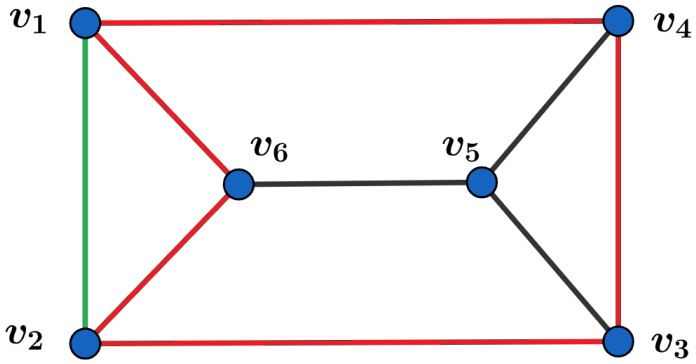

Figura 2: Cap induzido pelo conjunto de vértices $\{v_1, v_2, v_6, v_3, v_4\}$ com a corda v_1v_2.
Ilustração feita no software Geogebra.

Repetiremos o processo novamente, agora para a aresta v_1v_6, com o objetivo de ilustrar mais uma iteração deste algoritmo. A vizinhança de seus extremos é dada por:

$$N(v_1) = \{v_2, v_4, v_6\}, N(v_6) = \{v_1, v_2, v_5\}.$$

Note que v_2 é um vértice comum às vizinhanças de ambos os vértices. Assim, $v_2 \in N(v_1) \cap N(v_6)$. Note que $N(v_2) = \{v_1, v_3, v_6\}$.

O próximo passo do algoritmo é analisar se v_1 e v_6 possuem vizinhos em comum na componente $C = G \setminus ((N(v_1) \cap N(v_6)) \cup N(v_2))$. Dessa forma, temos

$$G \setminus \{v_2\} \cup \{v_1, v_3, v_6\} = \{v_4, v_5\}.$$

Note que, em $C = \{v_4, v_5\}$ tanto o vértice v_1 quanto o vértice v_6 possuem vizinhos. Desta forma, observe que o conjunto de vértices $\{v_1, v_2, v_6\} \cup \{v_4, v_5\}$ (destacado em vermelho), onde $v(P) = \{v_4, v_5\}$ (o caminho mais curto em C, visto que C é isomorfo a P_2) induz um cap com a corda v_1v_6, destacada em verde na Figura 3.

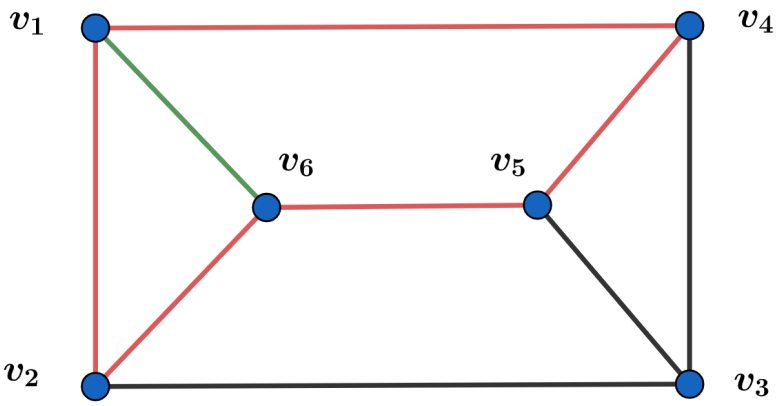

Figura 3: Cap induzido pelo conjunto de vértices $\{v_1, v_2, v_6, v_4, v_5\}$ com a corda v_1v_6.
Ilustração feita no software Geogebra.

Note que este algoritmo não nos retorna caps ao realizarmos o procedimento escolhendo arestas como v_1v_4, v_5v_6 e v_2v_3, pois estes extremos não possuem vértices em comum em suas vizinhanças.

Note que este grafo não é de Meyniel, pois pela definição dada acima, todo ciclo ímpar com pelo menos 5 vértices possui pelo menos duas cordas. Exibimos acima dois caps de G, isto é, ciclos com 5 vértices contendo apenas uma corda, contradizendo a definição de Meyniel para este grafo.

Referências

[1] Bienstock, D. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics 90*, 1 (1991), 85–92.

[2] C.M.H. de Figueiredo, J. Meidanis, C. M. *Coloração em Grafos*. XVI Jornada de Atualização em Informática, 1997.

[3] De Figueiredo, C. M., and Vušković, K. Recognition of quasi-meyniel graphs. *Discrete applied mathematics 113*, 2-3 (2001), 255–260.