Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada (DIMAp)
Programa de Pós-Graduação em Sistemas e Computação (PPgSC)

DIM0860 - Algoritmos e Estruturas de Dados em Grafos
Prof.: Elizabth Ferreira Gouvea Goldbarg

Exemplos de algoritmos em grafos

Componentes:
Fagner Morais dias (20191026321)

19 de março de 2020

Sumário

1 Algoritmo K-Best

1.1 Introdução

O problema do k-best consiste em, dado um grafo $G = (E, V)$, na geração de k árvores geradoras, onde seus pesos são $W_1 < W_2 < ... < W_k$, não existindo árvore geradora com menores pesos no grafo G.

Para a geração de árvores geradoras, existem uma quantidade considerável de algoritmos que, dado um grafo, temos como saída uma árvore geradora para o grafo. Para grafos que possuem pesos em suas arestas, é compreensível querer a árvore geradora que possui o menor peso possível no grafo (árvore geradora mínima). Para a construção de árvores geradoras mínimas a partir de um grafo temos, novamente, uma grande quantidade de algoritmos para sua construção, O algoritmos de Kruskal e Prim são bem conhecidos para a resolução deste problema. Para a geração das k-best árvores geradoras, dado um grafo, vamos utilizar o algoritmo proposto por (1), contudo há outros algoritmos que se propõem a solucionar esse problema, como (2; 3).

1.2 Algoritmo

O algoritmo de k-best, proposto por (1), funciona gerando partições e particionando partições de maneira a gerar uma sequencia de resultados, onde cada resultado é uma árvore geadora. Essa sequencia é realizada de maneira ordenada e de forma crescente. Uma partição é definida como um subconjunto não vazio do conjunto de todas as árvores geradoras de um grafo $G = (E, V)$, definido por:

$$\mathbb{P} = \{(i_1, j_1), ..., (i_r, j_r); (\overline{m_1, p_1}), ..., (\overline{m_l, p_l})\} \tag{1}$$

onde (i_1, j_1) representa arestas que estão de maneira obrigatória na árvore geradora, $(\overline{m_1, p_1})$ são as arestas que não fazem parte de árvore geradora e por fim temos as arestas que não estão representadas no conjunto $\mathbb{P}$, as quais são arestas opcionais na árvore. A árvore geradora mínima da partição $\mathbb{P}$ é definido como $s(\mathbb{P})$ e o seu custo é representado como $c[s(\mathbb{P})]$.

1.2.1 Funcionamento

Dado um grafo G contendo n vértices, o funcionamento do algoritmo é realizado em etapas, onde na etapa k será gerado a k-melhor árvore geradora mínima. O algoritmo realiza o seu processamento em dois estágio, o inicial e o estágio que será repetido a quantidade k vezes.

- estágio 0
  O conjunto de listas é igual a partição que contém todas as arestas do grafo G, ou seja:

$$s1 = \{(i_1, j_1), ..., (i_{n-1}, j_{n-1})\} \tag{2}$$

Particionando a partição $s1$, são criadas as partições $M_1, ..., M_{n-1}$ definidas como:

$$M_1 = \{\overline{(i_1, j_1)}\} \tag{3}$$
$$M_2 = \{(i_1, j_1), \overline{(i_2, j_2)}\} \tag{4}$$
$$... \tag{5}$$
$$M_{n-1} = \{(i_1, j_1), ..., (i_{n-2}, j_{n-2}), \overline{(i_2, j_2)}\} \tag{6}$$
$$\tag{7}$$

- estágio K
  Dado a lista de partições, é calculado então a árvore geradora mínima para cada partição contida na lista, juntamente com os seus custos associados.

1.2.2 Pseudocódigo

```
Input: Graph G(V,E) and weight function w
Output: Output_File (all spanning trees of G, sorted in order of increasing cost)
List = {A}
Calculate_MST (A)
while MST ≠ ∅ do
    Get partition P_s ∈ List that contains the smallest spanning tree
    Write MST of P_s to Output_File
    Remove P_s from List
    Partition(P_s).
```

Figura 1: Pseudocódigo para o algoritmo k-best

```
PROCEDURE PARTITION (P)
P_1 = P_2 = P;
for each edge i in P do
if i not included in P and not excluded from P then
    make i excluded from P_1;
    make i included in P_2;
    Calculate_MST (P_1);
    if Connected (P_1) then
        add P_1 to List;
    P_1 = P_2;
```

Figura 2: Pseudocódigo para a geração de partições

A complexidade do algoritmo apresentado por (1) é:

$$O(N|E|\log|E| + N^2) \tag{8}$$

onde N é o número de árvores geradoras, dado o grafo G e E, o número de arestas.

1.3 Exemplo

Iniciamos nosso exemplo com o grafo ponderado G, descrito na Figura 3.

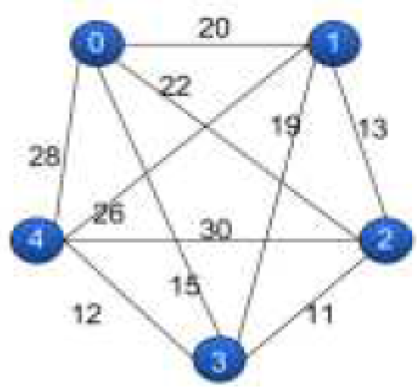

Figura 3: Grafo ponderado G

O primeiro passo é, a partir de G, gerar a sua árvore gerado mínima, tendo inicialmente todas as arestas classificadas como opcionais. Gerado a árvore geradora mínima, representado na Figura 4, com peso igual a 51.

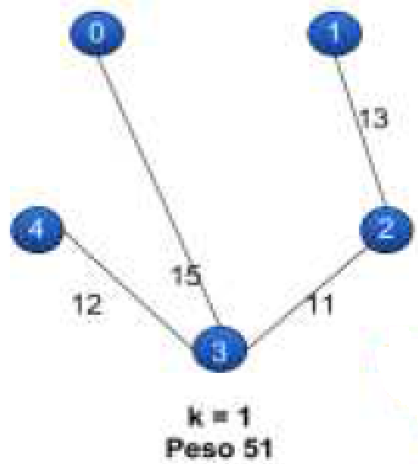

Figura 4: Árvore geradora mínima de G

Agora que temos a árvore geradora mínima (AGM), em relação ao grafo G, podemos iniciar a criação de partições. Para a criação de tais, temos que proibir arestas presentes na AGM. Note que a cada partição, uma aresta proibida se tornará obrigatória em outra partição. A representação das partições resultantes está apresentado na figura 5. Para cada partição também será realizada a identificação do peso de suas respectivas árvores geradoras mínimas.

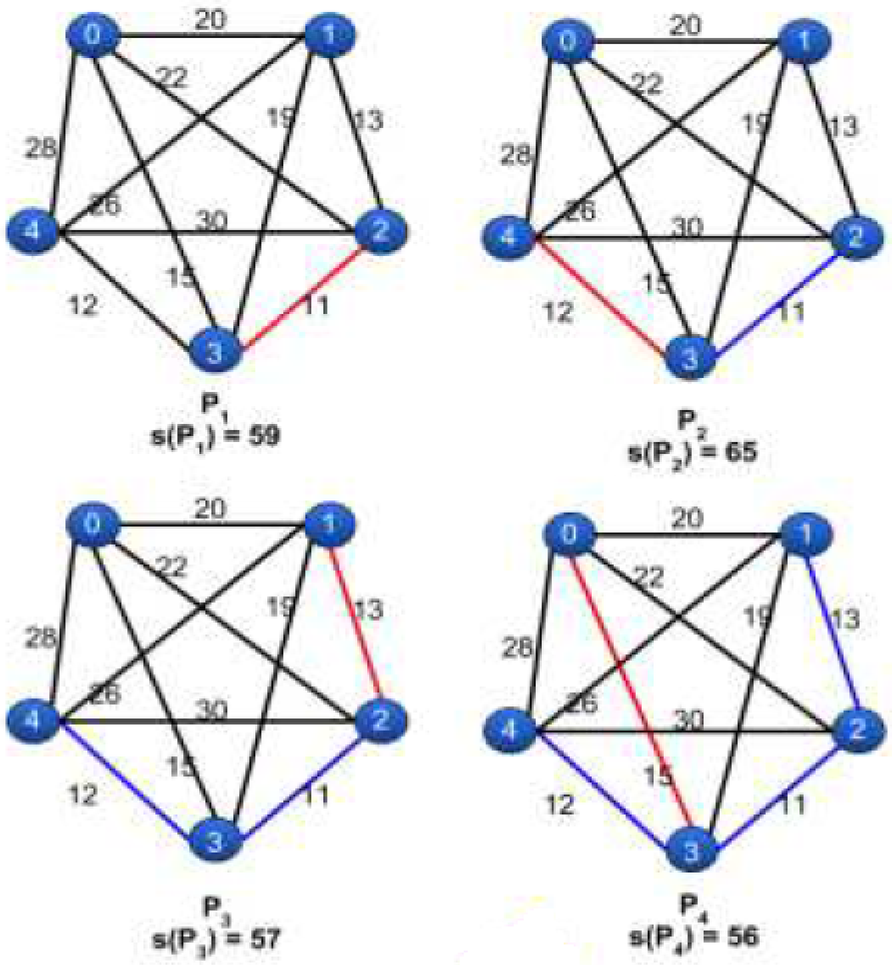

Figura 5: Lista de partições de G

Criada a lista de partições inicial, devemos pegar a partição que contém o menor peso, dado sua AGM, e a partir de sua AGM, gerar mais partições para acrescentar a lista de partições. Dado a lista de partições, apresentada na Figura 5, tempos o grafo que possui o peso de AGM igual a 56, Figura 6, a partir da sua AGM, será gerado novas partições que serão adicionadas a lita de partições, esse processo é mostrado na figura 7

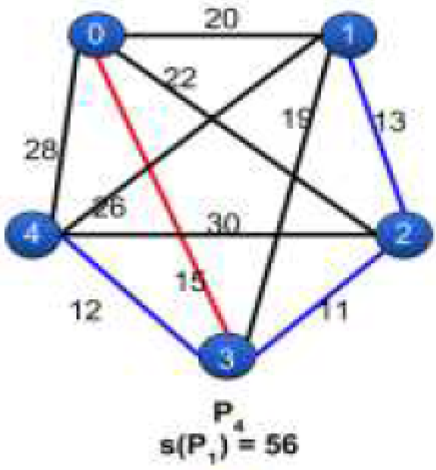

Figura 6: Grafo P_4

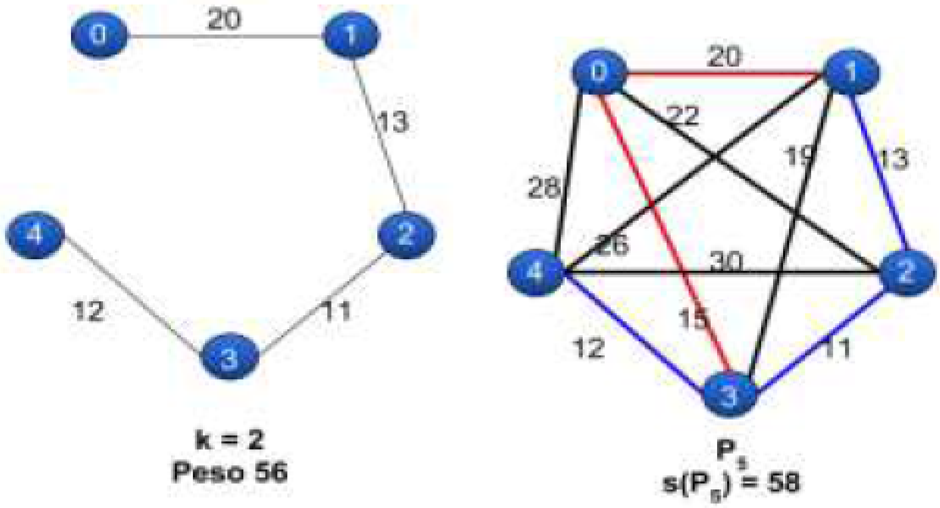

Figura 7: Grafo P_4

Esse processo será repetido até que a quantidade k de vezes seja alcançada ou que a lista de partições esteja vazia.

Referências

[1] K. SÃand G. K. Janssens, "An algorithm to generate all spanning trees of a graph in order of increasing cost," *Pesquisa Operacional*, vol. 25, pp. 219 – 229, 08 2005. pages 3, 5

[2] H. N. Gabow and E. W. Myers, "Finding all spanning trees of directed and undirected graphs," *SIAM Journal on Computing*, vol. 7, no. 3, pp. 280–287, 1978. pages 3

[3] T. Matsui, "An algorithm for finding all the spanning trees in undirected graphs," *METR93-08, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo*, vol. 16, pp. 237–252, 1993. pages 3