Centro Federal de Educação Tecnológica de Minas Gerais
Graduação em Engenharia da Computação

Vertex Cover

Otimização I

Professor: Amadeu Almeida

Alunos:
Antônio Augusto Diniz Sousa - 201712040146
Milena Delarete Drummond Marques - 201712040227
Sílvia Valadares da Silva Fonseca - 201712040286

04 de Dezembro de 2020

1 Descrição do problema

Dado um grafo G com v vértices e a arestas, uma cobertura de vértices (ou *Vertex Cover*) é um conjunto de vértices V em que cada aresta do grafo G é ligada a ao menos um dos vértices do conjunto V. Também pode ser definido como um conjunto de vértices cuja a remoção desconecta completamente um grafo[2].

O problema da cobertura de vértices mínima (ou *minimum vertex cover*) é um problema NP-completo muito conhecido na área de otimização e consiste em, dado um grafo G, encontrar a cobertura de vértices dele com o menor número de vértices possíveis.

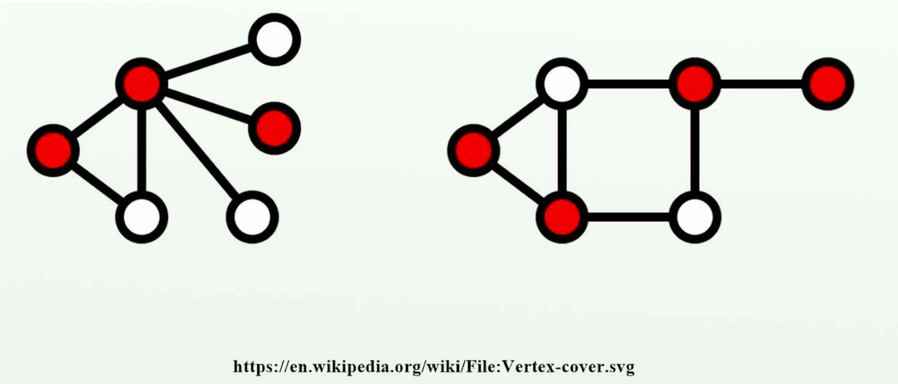

Figura 1: exemplos de grafos com sua coberturas de vértices destacadas em vermelho

Esse problema é usado em muitas situações no mundo real, sendo algumas possíveis aplicações: a instalação do mínimo de câmeras que cobrem todos os caminhos possíveis de um estabelecimento comercial[7]; monitoramento de uma rede de comunicação, com o mínimo de dispositivos em locais selecionados para monitorar o incidente de links de comunicação para o local onde ele está localizado[8]; em estudos de sequências de DNA[3]; ente outras.

Uma possível formulação matemática do problema, considerando os dados apresentados nas tabelas 1 e 2, é dada pelas equações (1) a (3):

Nome	Descrição
V	conjunto de vértices
$j \in V$	um vértice do grafo
A	conjunto de arestas
$(i,j) \in A$	uma aresta do grafo

Tabela 1: Constantes do Problema de Cobertura de Vértices.

Nome	Descrição
v_j	determina se um vértice j pertence ($v_j = 1$) ou não ($v_j = 0$) à cobertura de vértices

Tabela 2: Variáveis do Problema de Cobertura de Vértices.

Conjunto de variáveis: $\{v_j; \forall j \in V\}$

Função objetivo:

$$\min \sum_{j \in v} v_j \tag{1}$$

Sujeito à:

$$v_i = 1 \bigvee v_j = 1 \quad \forall \{i, j\} \in A \tag{2}$$

$$v_j \in \{0, 1\} \quad \forall j \in V \tag{3}$$

2 Revisão bibliográfica

O problema de cobertura de vértices é foco de muitos estudos na área de Pesquisa Operacional e muitos algoritmos já foram sugeridos para sua solução. Esses algoritmo podem ser divididos, em sua maioria, em dois grandes grupos: algoritmos exatos e algoritmos aproximados. Por ser um problema NP-completo, as soluções exatas possuem complexidade muito altas, como $O(2^w \cdot w^{3w+2} \cdot n)^5$, em que w é o comprimento da árvore e n é o número de vértices na decomposição da árvore, e $O(2,7606^k)^5$ ou $O(2,4882^k)^5$, em que k é o tamanho desejado da cobertura de vértices. Em razão dessas complexidades, esses algoritmos não conseguem processar grafos grandes e, para isso, os algoritmos aproximados são uma alternativa mais viável, chegando a complexidades baixas, como $O(log^2 n)^5$. As heurísticas desenvolvidas para esse trabalho são algoritmos aproximados.

3 Heurísticas selecionadas

A primeira heurística foi elaborada baseando-se em um algoritmo de lista[1], que recebe como entrada um grafo G e uma lista $L = u_1, ..., u_n$ cujo conteúdo é uma permutação dos vértices de G.

Nessa lista, foi considerado que u_i está à esquerda de u_j se $i < j$ e que, se u_i e u_j são vizinhos no grafo (ligados por uma aresta), u_i é vizinho esquerdo de u_j. Respectivamente, u_j está à direita de u_i se $j < i$ e, se u_i e u_j ligados por uma aresta no grafo, u_j é vizinho direito de u_i.

Considerando C o conjunto cobertura de vértices e que ele é inicialmente vazio, o algoritmo proposto caminha pela lista L da direita para a esquerda, ou seja, começando do último elemento em direção ao primeiro. Para cada vértice u encontrado na lista é feito a checagem: se u tem ao menos um vizinho **direito** que não pertence a C, então u é inserido no conjunto C. Caso contrário, o algoritmo segue para o próximo elemento da lista. Após ter passado por toda a lista, o algoritmo retorna C como o conjunto cobertura de vértices.

A lógica da segunda heurística consiste em começar pelo vértice ligado ao maior número de arestas, adicioná-lo ao conjunto cobertura e marcá-lo como visitado. Em seguida, todas as arestas ligadas a esse vértice são eliminadas e o algoritmo recalcula para todos os vértices não visitados, o número de arestas a que estão ligados e repete o procedimento até que não existam mais arestas para serem eliminadas.

As heurística implementadas podem ser encontradas no GitHub no link: `https://github.com/antonioaads/vertex-cover`.

4 Experimentos computacionais

4.1 Características do computador

O mesmo computador foi utilizado para fazer os testes de ambas as heurísticas para melhor comparação. Suas informações estão descritas na tabela abaixo:

Sistema operacional	Ubuntu 18.04.5
Memória	7,7 GB
Processador	Intel® Core™ i7-7500U CPU @ 2.70GHz x 4
Tipo de sistema	64 bits

Tabela 3: Característica do computador.

4.2 Execução

Para rodar as heurísticas implementadas, tendo o *python* instalado na máquina, basta rodar o arquivo **run.py** de dentro do diretório da heurística que desejar, passando como parâmetro um ou mais grafos descritos em arquivo.

```
python run.py graph.txt [...graphn.txt]
```

Após a execução, o código imprime, para cada grafo enviado, os vértices que foram escolhidos e o tempo que levou para execução.

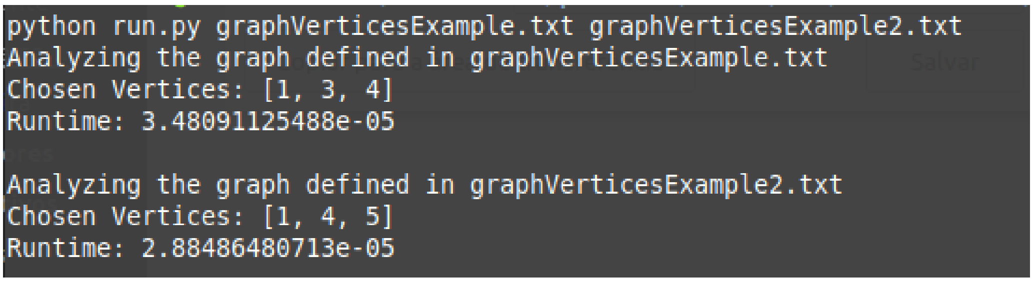

Figura 2: Exemplo de execução passando dois grafos

4.3 Experimentos

4.3.1 Grafos usados

Os grafos usados foram os mesmos para ambas as heurísticas e foram gerados aleatoriamente por um código em C++ que pode ser encontrado no GitHub no link `https://github.com/antonioaads/vertex-cover`. Foram testados dois grafos pequenos: A e B de 10 vértices cada, e dois grafos médios, C e D, de 50 vértices cada.

4.3.2 Resultados

			Tempo de processamento (em s)		Nº de vértices da cobertura	
Grafo	Nº de vértices	Nº de arestas	Heurística 1	Heurística 2	Heurística 1	Heurística 2
A	10	16	3,218651E-05	3,771782E-04	6	5
B	10	32	2,741814E-05	8,554459E-04	9	7
C	50	55	1,022816E-04	2,000570E-03	31	22
D	50	110	8,165836E-04	1,948357E-03	39	29

Figura 3: Tabela com os dados dos experimentos

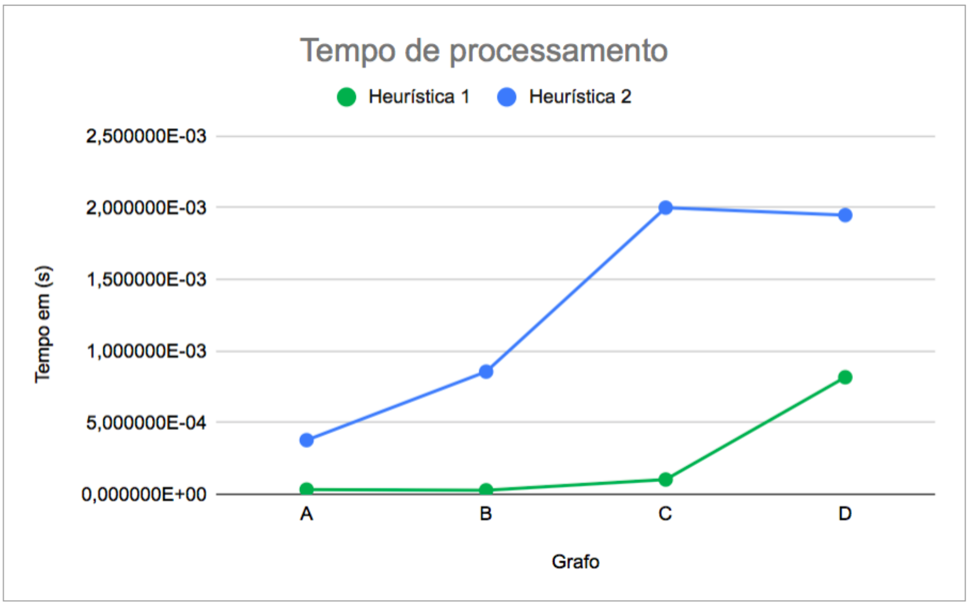

Figura 4: Gráfico comparando o tempo de execução das duas heurísticas

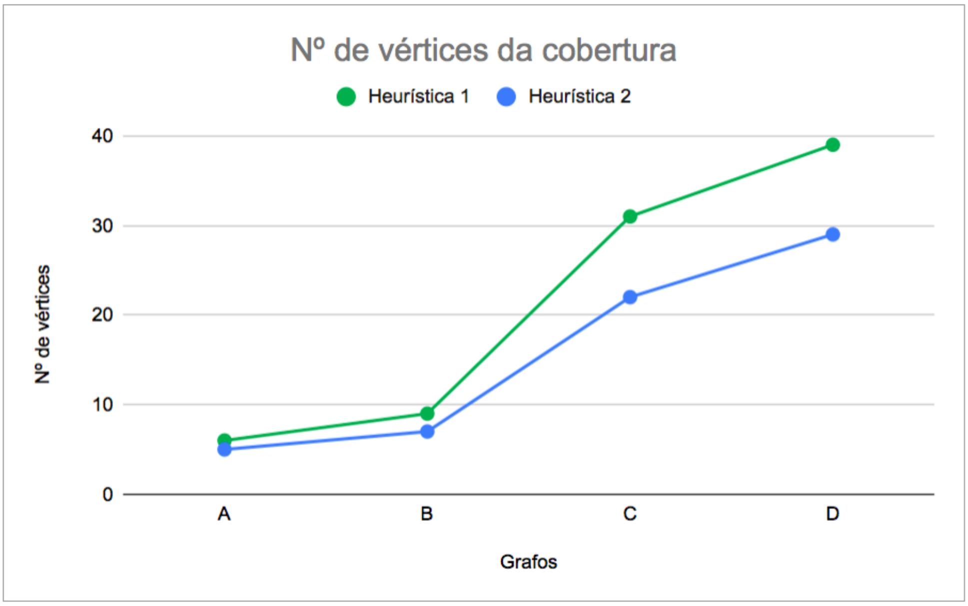

Figura 5: Gráfico comparando o número de vértice na cobertura encontrada pelas duas heurísticas

4.3.3 Conclusão

Com base nos resultados dos testes apresentados na seção anterior, a heurística 1 se mostrou mais rápida para todos os casos testados. Isso ocorre porque na heurística 2, para cada vértice adicionado à cobertura de vértices, é necessário recalcular os pesos dos vértices restantes. Enquanto na heurística 1, o grafo é necessário apenas percorrer a lista ordenada de vértices uma vez. Entretanto, a heurística 2 resultou em conjuntos com menores quantidades de elementos, ou seja, resultados mais próximo da solução ótima. Este resultado já era esperado, já que a heurística 1 é voltada para aplicações em que o grafo é recebido como uma lista cuja ordem não pode ser alterada, sendo a melhor heurística conhecida atualmente para casos em que há essa restrição. A heurística 2, por sua vez, ao adicionar ao conjunto solução o vértice com mais arestas ainda não cobertas, faz a melhor opção local, se aproximando mais da solução ótima para o algoritmo.

5 Referências

1. DELBOT, Francois LAFOREST, Christian. (2008). **A better list heuristic for vertex cover**. Disponível em: https://www.researchgate.net/publication/220114326_A_better_list_heuristic_for_vertex_cover. Acesso em: 30 nov. 2020.

2. SILVA, Mariana O. Da; GIMENEZ-LUGO, Gustavo A.; SILVA, Murilo V. G. Da. **VERTEX COVER IN COMPLEX NETWORKS**. Disponível em: http://www.inf.ufpr.br/murilo/public/VCCN.pdf. Acesso em: 1 dez. 2020.

3. BRAGA, Marília D. V. **Grafos de Sequências de DNA**. Disponível em: http://repositorio.unicamp.br/jspui/bitstream/REPOSIP/275909/1/Braga_MariliaDiasVieira_M.pdf. Acesso em: 2 dez. 2020.

4. ANGEL, Eric; CAMPIGOTTO, Romain; LAFOREST, Christian. **Implementation and Comparison of Heuristics for the Vertex Cover Problem on Huge Graphs**. Disponível em: https://hal.archives-ouvertes.fr/hal-00741605/document. Acesso em: 1 dez. 2020.

5. ZHANG, Yongfei; WU, Jun; ZHANG, Limin; ZHAO Peng; ZHOU, Junping; YIN Minghao. **An Efficient Heuristic Algorithm for Solving Connected Vertex Cover Problem**. Disponível em: https://www.hindawi.com/journals/mpe/2018/3935804/. Acesso em: 4 dez. 2020.

6. https://en.wikipedia.org/wiki/Vertex_cover. Acesso em: 1 dez. 2020.

7. https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/v-cover.html. Acesso em: 1 dez. 2020.

8. https://www.youtube.com/watch?v=0knYqPR6aOU&ab_channel=HenryAdams. Acesso em: 1 dez. 2020.

9. https://www.ic.unicamp.br/~fkm/disciplinas/mo418/2015s2/apresentacoes/gabriel-texto.pdf. Acesso em: 1 dez. 2020.

10. https://www.passeidireto.com/arquivo/18990781/cobertura-minima-de-vertices. Acesso em: 2 dez. 2020.

11. http://www.mat.uc.pt/~jsoares/teaching/oc/1112/RelatorioOC.pdf. Acesso em: 2 dez. 2020.