Modelagem para o Problema de Cobertura de Vértices Mínima

Álvaro Antônio Fonseca de Souza[1]
Daniel Carlos Hovadick Félix[1]
Guilherme Gonzaga Barbosa[1]

[1]Departamento de Ciência da Computação, Universidade Federal de Minas Gerais (UFMG)

Resumo. Este é um relatório que apresenta uma heurística para o Problema da Cobertura de Vértices Minima (PCVM). Este é um problema clássico em otimização, onde se deseja obter o número mínimo de vértices que cobrem todas as arestas de um grafo. Neste trabalho será apresentada um comparação de desempenho entre os resultados do solver GLPK e os resultados da Heurística implementada.

1. Introdução

O problema de cobertura de vértices minima é um problema clássico de otimização, onde em um grafo *G(V, A)*, quer se obter o número mínimo de vértices que estejam ligados a todas as arestas, minimizando a soma dos custos atribuídos aos vértices.

O problema de cobertura de vértices tem inúmeras aplicações, como a minimização do uso de câmeras que irá vigiar um conjunto de corredores de um ambiente. Dado que cada câmera pode vigiar mais de um corredor a partir das interseções destes, podemos atribuir o peso dos vértices a diversos fatores, como a qualidade de uma câmera que será utilizada. Em geral, o PCVM é aplicado onde se quer minimizar a quantidade de atributos e custos em uma solução de um problema de otimização. Esses atributos podem ser representados pelos vértices do grafo.

O PCVM faz parte da classe de problemas NP-Difíceis, o que significa que ainda não existem algoritmos de complexidade polinomial para resolvê-los na otimalidade. Desta forma este trabalho propõe a utilização de uma heurística baseada na ordenação do grau dos vértices do grafo e da seleção dos vértices em ordem decrescente de grau para o conjunto solução até que todas as arestas estejam cobertas.

2. Modelagem do Problema Cobertura de Vértice Minima

O modelo de Programação Linear para o PCVM pode ser escrito como:

$$
\begin{array}{lll}
\text{minimize} & \sum_{v \in V} c(v) x_v & \text{(minimiza o custo total)} \\
\text{sujeito a} & x_u + x_v \geq 1 \text{ para todo } \{u, v\} \in E & \text{(cobre todas as arestas do grafo)} \\
 & x_v \in \{0, 1\} \text{ para todo } v \in V. & \text{(cada vértice está na cobertura de vértice ou não)}
\end{array}
$$

3. Heurística proposta

A heurística proposta tem o seu funcionamento baseado no algoritmo descrito pelo pseudocódigo abaixo:

```
ALGORITMO
    G ← {V, A}
    S ← ∅
    ENQUANTO A ≠ ∅ FAÇA
        v' ← vértice de maior grau em G
        A ← A - {arestas incidentes à v'}
        S ← S + {v'}
    FIM ENQUANTO
    RETORNE S
FIM
```

O algoritmo, ao ler o arquivo de entrada, cria um arranjo de vértices (cada índice contém o identificador, o grau e o peso referente àquele vértice); e uma lista de arestas (contendo os identificadores dos vértices conectados por aquela aresta). A heurística desenvolvida trabalha analisando o grau dos vértices para definir àqueles que serão acrescentados ao conjunto-solução. Em nenhum momento, será levado em conta o custo de cada vértice.

Dessa forma, o algoritmo escolhe o vértice de maior grau e o adiciona à solução. A seguir, as arestas incidentes sobre esse vértice são retiradas da lista de arestas e o grau dos vértices restantes no grafo são atualizados (ou seja, o grau será definido considerando apenas as arestas ainda não cobertas pela solução parcial). Esse processo se repete até que não reste mais arestas na lista, o que indica que todas as arestas já se encontram cobertas. Nesse ponto, temos a solução final.

4. Implementação

O programa foi escrito utilizando a linguagem de programação C. Detalhes da implementação, compilação e execução do programa que implementa a heurística são descritos nesta seção.

4.1. Representação de uma solução da cobertura

O custo de uma solução para o PCVM é armazenado em um acumulador para os custos de vértices selecionados a cada passo. Uma solução pode ser representada pelos vértices em preto no grafo abaixo:

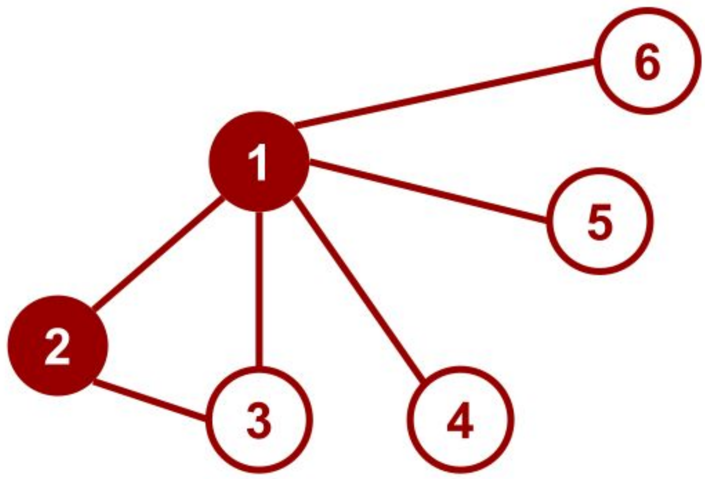

Figura 1 - Exemplo de solução para o PCVM.

4.2. Compilação e Execução do Programa

A compilação do código é feita através do utilitário *make*. Para isso dentro do diretório onde se encontra o arquivo *Makefile*:

```
$ make
```

O arquivo executável com a heurística fica disponível no diretório *bin*, e para executá-lo forneça os parâmetros descritos a seguir:

```
$  bin/tp.exec [arquivo_de_entrada] [arquivo_de_saída]
```

5. Modelagem para o GLK

Para a modelagem do problema para o GLPK, foi utilizada a linguagem GPML. O modelo é definido pelo código abaixo:

```
param N;
set A, dimen 2;
param c{i in 1..N};
var x{i in 1..N}, binary;
s.t. cov{(i,j) in A}: x[i] + x[j] >= 1;
minimize z: sum{i in 1..N} c[i] * x[i];
```

As instâncias do problema são do tipo:

```
param N := 3;
set A :=
1, 2
1, 3
2, 3;
```

```
param c :=
1 1
2 4
3 2;
```

Onde N é o número de vértices, A é o conjunto de arestas e c é o vetor de custos associados aos vértices.

6. Testes

Os testes foram realizados para grafos contendo entre 10 e 100 vértices. Essa decisão foi tomada em vista do fato de o solver necessitar de um tempo muito grande para solucionar grafos maiores que esses. Tornado a comparação com a heurística mais problemática.

As instâncias foram geradas através de um *script*, que determina de forma aleatória o número de arestas do grafo gerado. Todo os grafos são não-ponderados e não-direcionados. Todos os testes foram realizados em uma máquina com processador Intel Atom N570 e 2 GB de memória RAM, utilizando o sistema operacional Ubuntu 14.04.

À seguir, uma tabela e os gráficos contendo os resultados obtidos por ambas as ferramentas para as instâncias de teste. Para efeito de comparação, analisaremos o tempo de execução, o número de vértices selecionados e o custo total da solução encontrada.

Arestas	Vértices	Heurística			Solver		
		Tempo (ms)	Vértices na solução	Custo	Tempo (ms)	Vértices na solução	Custo
31	10	4	7	364	16	7	364
149	20	5	17	776	43	17	677
157	30	5	22	1198	122	24	1136
84	40	5	22	1009	25	25	849
1189	50	8	48	2609	907	48	2538
90	60	5	30	1209	28	31	1119
377	70	6	52	2312	2651	52	2108
2095	80	12	75	3607	11344	75	3443
747	90	5	73	3471	49234	74	3153
3825	100	15	96	4362	27323	95	4240

Tabela 1 - Comparação Heurística x GLPK.

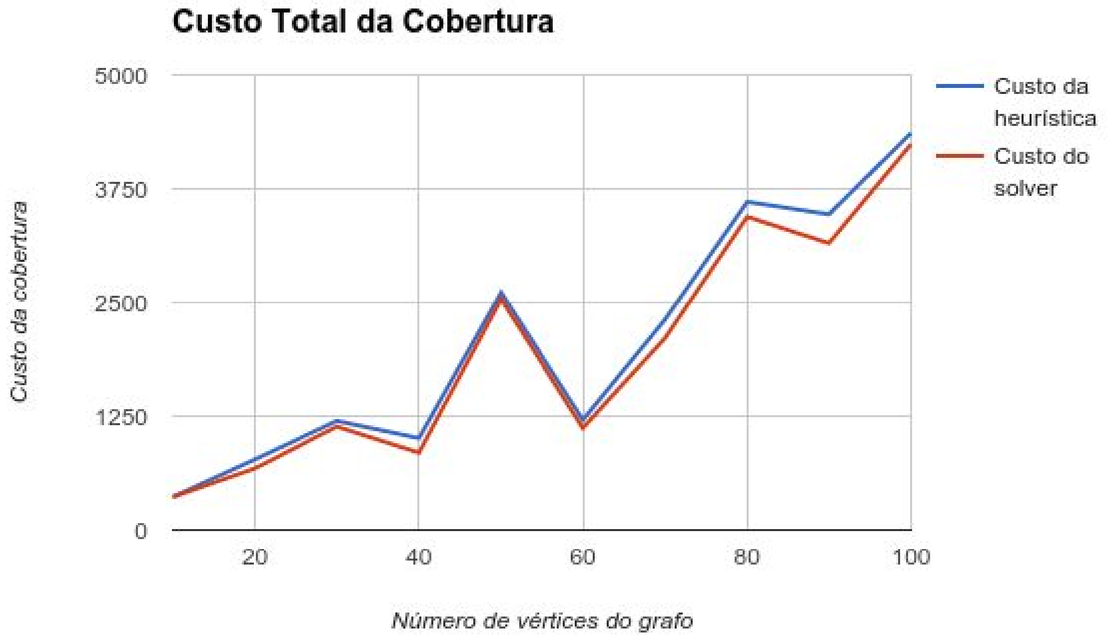

Gráfico 1 - Custo total.

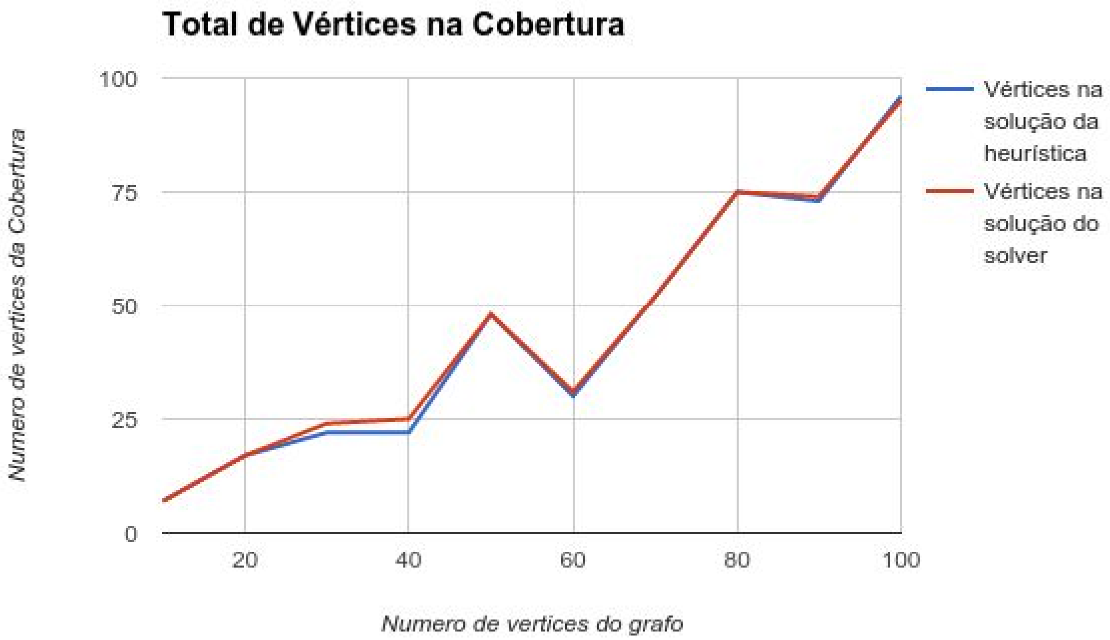

Gráfico 2 - Número de vértices selecionados

Gráfico 3 - Tempo de execução.

7. Conclusão

No geral, a heurística teve um bom desempenho em relação ao número de vértices acrescentados à solução final e em relação ao tempo de execução, em média 0,007 segundos contra 9,2 segundos para o solver.

Entretanto, ela sempre é batida no fator custo - uma vez que o algoritmo não considera os pesos dos vértices. A heurística implementada mostra-se mais recomendável para instâncias onde os custos dos vértices abrangem um intervalo pequeno ou, de preferência, sejam exatamente os mesmos para todos os nós.

8. Referências

Ziviani, N. Projeto de Algoritmos Com Implementações em Pascal e C, Pioneira Thomson Learning, 2004, segunda edição.

Vertex Cover - Wikipedia, The Free Enciclopedia: Acesso em 07/05/2015. Disponível em: https://en.wikipedia.org/wiki/Vertex_cover