UNIVERSIDADE FEDERAL DE MINAS GERAIS
Graduação em Engenharia de Sistemas
Algoritmos e Estruturas de Dados III - Trabalho Prático 2
Uaibucks
Matheus Silva Araujo - 2013066265

1 Introdução

No Trabalho Prático 2 da disciplina de Algoritmos e Estruturas de Dados III, nosso velho conhecido Nubby abriu uma startup de consultoria e tem a famosa cafeteria *Uaibucks* como cliente. Essa cafeteria deseja instalar-se em uma nova cidade, e para isso impôs algumas restrições.

A cidade possui N esquinas, duas esquinas são vizinhas se há uma rua que as interligue. Cada esquina possui uma demanda associada $d_i \in \mathbb{N}$. A empresa deseja abrir cafeterias em um subconjunto de esquinas, tais que duas cafeterias não estejam em esquinas vizinhas e a demanda esperada seja máxima.

Para este trabalho, é necessário:

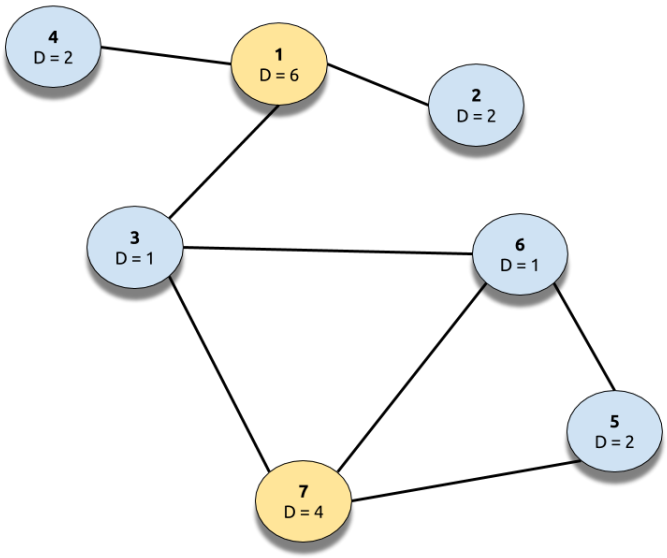
- Modelar o problema utilizando teoria de grafos;
- Provar que o problema pode ser modelado como um problema de decisão;
- Provar que o problema de decisão é um problema NP-Complemento;
- Implementar um algoritmo exato que resolva o problema;
- Implementar uma heurística para o problema.

2 Grafo

O problema da *Uaibucks* pode ser modelado através de um grafo, $G = (V, A)$, não direcionado, ponderado nos vértices.

Para a entrada de exemplo do enunciado, o grafo é apresentado na Figura 1.

Figura 1: Modelagem em Grafo



Os critérios estabelecidos pela *Uaibucks* definem o que é conhecido em Teoria dos Grafos como *Conjunto Dominante*.

Um conjunto dominante D em um grafo é um conjunto tal que qualquer vértice v do grafo ou está no conjunto D ou é adjacente a pelo menos um vértice contido no conjunto.

3 Problema de decisão

O problema apresentado é essencialmente um problema de otimização. Deseja-se encontrar um *Conjunto Dominante* que tenha demanda total máxima.

No entanto, é possível transformá-lo em um problema de decisão do tipo: caso a demanda máxima encontrada seja maior que um valor k, a empresa irá se instalar na cidade; caso contrário, não irá se instalar.

4 NP-Completude

O problema de encontrar um *Conjunto Dominante*, CD, é um problema de decisão *NP-Completo*.

Para provar isso, o primeiro passo é mostrar um algoritmo determinista polinomial que verifica o conjunto dominante, Algoritmo 1.

Algoritmo 1 Verifica CD

```
1: if |S| > n then
2:     return FALSE
3: for i = 1 to |V| do
4:     if V[i] ∉ S then
5:         adj = FALSE
6:         for j in S do
7:             if V[i], j ∈ A then
8:                 adj = TRUE
9:         if adj = FALSE then
10:            return FALSE
11: return TRUE
```

O próximo passo é mostrar que existe um problema $\pi \in$ *NP-Completo* que possa ser transformado no *Problema do Conjunto Dominante*.

Para isso, será usado o problema da *Cobertura de Vértices*, CV, conhecidamente *NP-Completo*.

Inicialmente, deve-se converter o grafo $G(V, A)$ com um valor k do problema CV, em um grafo $G'(V', A')$ para o problema CD com um valor n.

Define-se n e k.

Para construir G' em tempo polinomial:

- Todo vértice de G também é vértice de G'.
- Toda aresta de G também é aresta de G'.
- Para cada aresta (i,j) $\in$ A, cria-se um vértice z_{ij} em G e adicionam-se duas novas arestas (i, z_{ij}) e (z_{ij}, j).

Se G tem cobertura de vértice menor ou igual k, G' tem conjunto dominante de tamanho menor ou igual a n.

Se G' tem CD de tamanho menor ou igual a n, G terá CV de tamanho menor ou igual a k.

5 Algoritmo Exato

O Algoritmo Exato implementado para resolver o problema tem a seguinte estrutura:

- Gerar todos os subconjuntos possíveis do grafo, complexidade $O(2^n)$, onde n é o número de vértices do grafo;
- Para cada subconjunto validar se é um *Conjunto Dominante*, complexidade $O(n)$;
- Verificar se a demanda do conjunto é máxima, $O(1)$.

A complexidade final do algoritmo é $O(n \cdot 2^n)$.

6 Heurística

A Heurística para resolver o problema foi implementada com a seguinte estrutura:

- Ordenar os vértices do grafo (esquinas) pelo peso (demanda), em ordem decrescente. Em caso de empate, pelo grau do vértice (quantidade de esquinas vizinhas), ordem crescente. Complexidade $O(n \cdot log(n))$
- Com a lista ordenada, selecionar as esquinas com maior demanda que não são vizinhas das esquinas selecionadas anteriormente. Complexidade $O(n)$.

A complexidade final da heurística é $O(n + n \cdot log(n))$.

7 Resultados

Foram feitos testes para as duas implementações com os testes fornecidos. Apesar das diferentes complexidades das duas implementação, não foram observadas grandes diferenças nos tempos de execução, em função dos tamanhos das entradas.

Os resultados das execuções para as duas implementações são apresentados na Tabela 1.

O valor de acerto médio da Heurística foi de 88.62%, com desvio padrão de 19.00%. Em quatro dos 10 testes, a heurística encontrou o valor máximo; e em apenas dois valores abaixo de 60%, que a reprovariam.

Tabela 1: Resultados

Caso de teste	**N**	**M**	**Resultado exato**	**Resultado heurística**	**%**
in_1	4	4	118	100	84.75
in_2	5	4	168	121	72.02
in_3	6	14	170	170	100.00
in_4	7	8	10	10	100.00
in_5	8	2	304	304	100.00
in_6	9	26	139	82	58.99
in_7	6	11	30	25	83.33
in_8	10	17	265	261	98.49
in_9	10	16	312	312	100.00
in_10	10	38	198	87	43.94

8 Conclusão

Durante a modelagem e implementação do trabalho foi possível reforçar os conceitos da *Teoria de Grafos*, de *NP-Completude* e também de *Heurísticas*.

A diferença de complexidade temporal entre a heurística implementada e o algoritmo exato foi grande, caindo de uma complexidade exponencial para uma linear. O índice de acerto da heurística - apesar de ser uma ideia muito simples - foi satisfatório para os testes realizados.