

Trabalho Prático MC548 - Projeto e Análise de Algoritmos II

Alunos: Thiago Barros Rodrigues Costa - RA017447
Ribamar Santarosa de Sousa - RA 017209

Professor: Zanoni Dias

1 Problema -1)

Três informações definem completamente as restrições do enunciado do problema:

1. $|U| = \lfloor \frac{|V|}{2} \rfloor$, $U \subset V$.
2. $\delta(U) = (i,j) \in E$ onde $i \in U$ e $j \notin U$
3. $|\delta(U)|$ deve ser mínimo.

Os dados fornecidos são: n , inteiro com o número de vértices do grafo; *arestas*, array $n \times n$ com as arestas do grafo. A valor variável *nomearq*, que contém o nome do arquivo de inicialização, deve ser fornecido pelo usuário.

Variáveis do modelo: Na solução sugerida foram escolhidas as seguintes variáveis de modelo:

- *U*: Array binário de tamanho n representando o subconjunto U definido na descrição do problema. $U(i) = 1$ se, e somente se, o vértice i pertencer ao subconjunto U .
- *corte*: Array binário de tamanho $n \times n$ usado na definição das arestas do equicorte. $corte(i,j) = 1$ se, e somente se, $i \in U$ e $j \notin U$.

Restrições: As restrições sobre as variáveis foram:

- $(\sum_i^n U(i)) = \lfloor \frac{|V|}{2} \rfloor$): Trata diretamente da restrição 1. do problema apontada acima. Garante que U tenha o tamanho desejado (note que o somatório à esquerda vale exatamente o tamanho de U).

- $(U(j) - U(i) + corte(i, j) \geq 0)$: Sabemos que o equicorte é formado por $\delta(U) = \{(i, j) \in E | i \in U \text{ e } j \notin U\}$, *i.e.*, usando a definição de U , $\delta(U) = \{(i, j) \in E | U(i) = 1 \text{ e } U(j) = 0\}$. Queremos que a variável $corte(i, j)$ valha 1 se, e somente se, $U(i) = 1$ e $U(j) = 0$.

Note que a restrição acima nos garante que se $U(i) = 1$ e $U(j) = 0$ então $corte(i, j) = 1$. Veremos a seguir que, na solução minimizada $corte(i, j) = 0$ em todos os outros casos. Essa restrição será importante para a determinação das arestas que estão no equicorte.

Função Objetivo: Queremos minimizar $|\delta(U)|$, que é o conjunto de arestas no equicorte. Mas pelas restrições impostas a $corte(i, j)$, (i, j) está no equicorte se $corte(i, j) = 1$. Logo, (i, j) é uma aresta do equicorte se (i, j) for uma aresta $(i, e, aresta(i, j) = 1 \text{ ou } aresta(j, i) = 1)$ e $corte(i, j) = 1$. Podemos definir nossa função objetivo como:

$$\sum_{i,j}^n corte(i, j),$$

onde i e j variam sobre todas as arestas do grafo.

Note que $corte(i, j)$, pelas restrições, não precisa ser 0 se $U(i) \neq 1$ ou $U(j) \neq 0$. Mas como estamos tratando de uma minimização, na solução minimizada teremos todos esses valores valendo 0.

2 Problema -2)

O enunciado do problema pode ser resumido por:

1. Existem n setores e m possíveis locais para instalação de depósitos.
2. O custo da instalação do depósito j é $custo_fixo(j)$
3. Cada setor deve estar alocado a um único depósito.
4. $distancia(i, j)$ é a distância entre o setor i e o depósito j .

5. $custo_km$ é o custo por km.
6. O depósito j tem capacidade $capacidade(j)$.
7. A produção de lixo do setor i é $req(i)$.
8. Queremos minimizar os custos.

Os dados fornecidos são: n , m , $custo_km$, $capacidade$, req , $custo_fixo$ e $distancia$.

Variáveis do modelo: Na solução sugerida foram escolhidas as seguintes variáveis de modelo:

- **coleta:** Array binário de tamanho $n \times m$. $coleta(i, j) = 1$ se, e somente se, um depósito j foi alocado para um setor i .
- **deposito:** Array binário de tamanho m . $deposito(i)$ vale 1 caso o depósito i tenha alocado pra algum setor. Será usado no calculo do custo total junto com a variável $custo_fixo$.

Restrições: As restrições sobre as variáveis foram:

- $(\sum_{j=1}^n coleta(i, j) = 1, \forall i)$: Trata da questão 3. da acima na definição do problema, na qual somente um depósito pode ser alocado para cada setor.
- $(\sum_{i=1}^m coleta(i, j) * req(i) \leq capacidade(j), \forall j)$: Verifica se a quantidade de material alocada pra determinado deposito é compatível com sua capacidade. Para um dado depósito, a soma das produções de todos os setores alocados a ele deve ser menor ou igual que sua capacidade.
- $(\sum_{j=1}^n coleta(i, j) \leq deposito(j) * m, \forall j)$: Note que se $deposito(j) = 1$ a desigualdade acima sempre é satisfeita, mas quando $\sum_{j=1}^n coleta(i, j) > 0$ ela não $deposito(j)$ pode valer 0. Isso garante que $deposito(j)$ vale 1 sempre que o depósito j foi usado. Obs: Como estaremos usando uma minimização de um somatório com valores positivos que dependem de $deposito(j)$, o valor de tal variável será 0 sempre que o depósito j não for usado.

Função Objetivo: Queremos minimizar o custo total anual. Então devemos considerar o custo fixo de instalação dos depósitos ($custo_fixo$), além dos custos com transporte. O custo com transporte é dado por:

$$\sum_{i=1, j=1}^{i=n, j=m} coleta(i, j) * distancia(i, j) * custo_km * req(i) * 1000.$$

O custo fixo de instalação dos depósitos vale:

$$\sum_{j=1}^m custo_fixo(j) * deposito(j).$$

A função objetivo total vale então:

$$Objetivo = \sum_{i,j=1}^{i=n,j=m} coleta(i,j) * distancia(i,j) * custo_km * req(i) * 1000 + \sum_{j=1}^m custo_fixo(j) * deposito(j).$$

3 Problema -3a)

Podemos entender o problema com o seguinte:

1. Existem duas cadeias de restaurantes.
2. São n possíveis locais de instalação dos mesmos.
3. Em cada local só pode ser instalado um restaurante.
4. Não se pode instalar restaurantes da mesma cadeia em locais que distam menos de 5km entre si.
5. O lucro mensal com a instalação de uma restaurante da cadeia i no local j é dado por $lucro(i,j)$

Os dados fornecidos são: n , $lucro$, $Menos5$ ($Menos5(i,j)$ vale 1 se os locais i e j distam menos que 5km).

Variáveis do modelo: A única variável utilizada foi: - *restaurante*: Array binário de tamanho $2 \times n$. $restaurante(i,j) = 1$ se, e somente se, um restaurante da cadeia i foi instalado no local j

Restrições: As restrições sobre as variáveis foram:

- ($restaurante(1, i) + restaurante(2, i) \leq 1, \forall i$): De acordo com 3., em um determinado local só pode ser instalado restaurante de uma cadeia. A restrição trata desse problema.

- ($restaurante(1, i) + restaurante(1, j) \leq 1, \forall i, j$ tal que $Menos5(i, j) = 1$): Trata do problema de que dois restaurantes da cadeia 1 não podem ser instalados a uma distância menor que 5km.

$restaurante(2, i) + restaurante(2, j) \leq 1, \forall i, j$ tal que $Menos5(i, j) = 1$): Semelhante à restrição anterior, sendo que para a cadeia 2.

Função Objetivo: Queremos maximizar os lucros total, que é dado pela soma dos lucros com a cadeia 1 e a soma dos lucros com a cadeia 2. Tal valor pode ser calculado com a seguir:

$$Objetivo = \sum_{i=1}^n lucro(i, 1) * distancia(1, i) + lucro(2, 1) * distancia(2, i).$$

4 Problema -3b)

A parte b) do problema 3 é semelhante à parte a) sendo que é imposta uma restrição adicional de que se instalar um restaurante de uma cadeia num local então obrigatoriamente irá instalar um restaurante da outra cadeia a uma distância não superior a 5km.

As variáveis são as mesmas, assim como a função objetivo. As restrições da parte a) continuam sendo usadas, sendo que ainda adicionamos:

Restrições Acrescentadas:

- ($restaurante(1, i) \leq \sum_{j=1}^n Menos5(i, j) * restaurante(2, j), \forall i$): Veja que nesse caso se um restaurante da cadeia 1 for instalado, obrigatoriamente deverá ser instalado um restaurante da cadeia 2 a menos de 5km.

$restaurante(2, i) \leq \sum_{j=1}^n Menos5(i, j) * restaurante(1, j), \forall i$: Restrição análoga à anterior, sendo que para cadeia 2.

5 Problema -4)

O problema pode ser entendido da seguinte forma:

1. Um tanque de combustível possui n compartimentos.

2. A capacidade do compartimento i é de $capacidade(i)$.
3. Existem m tipos de combustíveis.
4. A demanda de cada combustível é dada pelo vetor $demanda$.
5. Não se pode misturar dois tipos de combustível em um mesmo compartimento.
6. Deseja-se minimizar a quantidade de combustível que não pode ser transportado.

Os dados fornecidos são: n , m , $capacidade()$ e $demanda()$.

Variáveis do modelo:

- **compartimento:** Array binário de tamanho $n \times m$. $compartimento(i, j) = 1$ se, e somente se no compartimento i foi usado combustível j .
- **sobrou:** Array de tamanho m com a quantidade de combustível que não pôde ser transportado para cada tipo.

Restrições:

As restrições sobre as variáveis foram:

- $(\sum_{j=1}^m compartimento(i, j) \leq 1, \forall i)$: Essa restrição garante que somente um tipo de combustível usado em para cada compartimento.
- $(demanda(i) - \sum_{j=1}^n compartimento(j, i) * capacidade(j) \leq sobrou(i), \forall i \leq m)$: Garante que a variável que o valor da variável $sobrou$ é pelo menos igual à quantidade de combustível que realmente deixou de ser transportado. Na verdade $sobrou$ vai valer exatamente essa quantidade sempre que ela for positiva
- $(sobrou(i) \geq 0, \forall i \leq m)$: Se não todo combustível pôde ser transportado então $sobrou$ deve valer 0

Função Objetivo:

Como queremos minimizar a quantidade de combustível que não foi transportado, podemos usar os valores da variável $sobrou$. Note que, com a função de minimização, $sobrou$ vai valer 0 sempre que não sobrou nada e vai valer a quantidade de combustível não transportado sempre que esse valor for positivo. Logo a função objetivo é:

$$\sum_{i=1}^m sobrou(i)$$