

Gile Peter Pereira Inêz RA: 016182
Pablo Cobucci Leite RA: 017018

Trabalho de MC-548

Professor : Zanoni Dias

Problema 1 -> Equicorte Mínimo

Formulação

Variáveis

- n – Lida do arquivo de entrada com o número de vértices. (inteiro)
- $arestas$ – Lida do arquivo de entrada, é uma matriz que recebe as arestas do grafo. (inteiro)
- $conjunto$ – Array de 1 até n de variáveis do modelo que contém os vértices de um lado do corte. (binário)
- $corte$ – Matriz $n \times n$ de variáveis do modelo que guarda as arestas do corte. (binário)

Objetivo

$$\text{Min} \quad \sum_{i=1}^n \sum_{j=1}^j corte(i, j)$$

Restrições

Para todo i de um até n faça
 Para todo j de um até n faça
 Se $arestas(i, j) == 1$ então #A aresta deve ter vértice nos 2 conjuntos.
 $corte(i, j) = | conjunto(i) - conjunto(j) |$
 Fim-Se
 Fim-Para
Fim-Para

$$n \text{ div } 2 = \sum_{i=1}^n conjunto(i) \quad \text{O conjunto deve ter a metade dos vértices.}$$

Problema 2 -> Coleta de lixo.

Formulação

Variáveis

- n – Lida do arquivo de entrada com o número de setores da cidade. (inteiro)
- m – Lida do arquivo de entrada com o número de depósitos de lixo. (inteiro)
- custo_km – Lida do arquivo de entrada com os custos por quilômetro. (real)
- capacidade – Lida do arquivo de entrada com o array de capacidade de cada depósito. (inteiro)
- req – Lida do arquivo de entrada com o array de demandas de cada setor. (inteiro)
- custo_fixo – Lida do arquivo de entrada com o array de custo de instalação de cada depósito. (real)
- distancia – Lida do arquivo de entrada com a matriz de distância entre setores e depósitos. (real)
- deposito – Array de variáveis do modelo verifica instalação do depósito. (binário)
- atende – Matriz de variáveis do modelo que verifica se o depósito atende o setor. (binário)

Objetivo

$$\text{Min} \quad \sum_{j=1}^m \text{deposito}(j) * \text{custo_fixo}(j) + \sum_{i=1}^n \sum_{j=1}^m \text{req}(i) * \text{atende}(i, j) * \text{distancia}(i, j) * \text{custo_km} * 1000$$

Obs.: É multiplicado por 1000 pois a capacidade está em milhares de metros cúbicos.

Restrições

```
Para todo i de 1 até n faça
    Para todo j de 1 até m faça
        Se existe distancia(i,j) então
            deposito(j) ≥ atende(i,j)
        Fim-Se
    Fim-Para
Fim-Para
Para todo i de 1 até n faça
     $\sum_{j=1}^m \text{atende}(i, j) = 1$ 
Fim-Para
Para todo j de 1 até m faça
     $\sum_{i=1}^n \text{req}(i) * \text{atende}(i, j) \leq \text{capacidade}(j)$ 
Fim-Para
```

Problema 3 a -> Problema de restaurantes com estratégia 'a'.

Formulação

Variáveis

- n – Lida do arquivo de entrada com o Número de possíveis locais para instalação dos restaurantes. (inteiro)
- $lucro$ – Lida do arquivo de entrada com a matriz que indica o lucro esperado por cadeia em cada local. (real)
- $Menos5$ – Lida do arquivo de entrada com a matriz que indica se a distância entre os pontos é menor que 5km. (inteiro)
- $cadeia1$ – Array de variáveis do modelo que indica se o restaurante da cadeia 1 deve ser instalado no ponto. (binário)
- $cadeia2$ - Array de variáveis do modelo que indica se o restaurante da cadeia 2 deve ser instalado no ponto. (binário)

Objetivo

$$\text{Max} \quad \sum_{j=1}^n cadeia1(j) * lucro(j,1) + \sum_{j=1}^n cadeia2(j) * lucro(j,2)$$

Restrições

Para todo i de 1 até n faça

Para todo j de 1 até m faça

#Garanto que vai haver no 0 ou 1 restaurante no ponto i .

$$cadeia1(i) + cadeia2(i) \leq 1$$

#Garanto que não haverá 2 restaurantes da cadeia1 distantes menos #que 5Km.

$$cadeia1(i) + Menos5(i,j) + cadeia1(j) \leq 2$$

#Garanto que não haverá 2 restaurantes da cadeia2 distantes menos #que 5Km.

$$cadeia2(i) + Menos5(i,j) + cadeia2(j) \leq 2$$

Fim-Para

Fim-Para

Problema 3 b -> Problema de restaurantes com estratégia 'b'.

Formulação

Variáveis

- n – Lida do arquivo de entrada com o Número de possíveis locais para instalação dos restaurantes. (inteiro)
- $lucro$ – Lida do arquivo de entrada com a matriz que indica o lucro esperado por cadeia em cada local. (real)
- $Menos5$ – Lida do arquivo de entrada com a matriz que indica se a distância entre os pontos é menor que 5km. (inteiro)
- $cadeia1$ – Array de variáveis do modelo que indica se o restaurante da cadeia 1 deve ser instalado no ponto. (binário)
- $cadeia2$ - Array de variáveis do modelo que indica se o restaurante da cadeia 2 deve ser instalado no ponto. (binário)

Objetivo

$$\text{Max} \quad \sum_{j=1}^n cadeia1(j) * lucro(j,1) + \sum_{j=1}^n cadeia2(j) * lucro(j,2)$$

Restrições

Para todo i de 1 até n faça

Para todo j de 1 até m faça

#Garanto que vai haver no 0 ou 1 restaurante no ponto i .

$$cadeia1(i) + cadeia2(i) \leq 1$$

#Garanto que não haverá 2 restaurantes da cadeia1 distantes menos #que 5Km.

$$cadeia1(i) + Menos5(i,j) + cadeia1(j) \leq 2$$

#Garanto que não haverá 2 restaurantes da cadeia2 distantes menos #que 5Km.

$$cadeia2(i) + Menos5(i,j) + cadeia2(j) \leq 2$$

Fim-Para

Fim-Para

Para todo i de 1 até n faça

#Garanto que no raio de 5km de um restaurante na cadeia1 haverá pelo menos

#um restaurante da cadeia2

$$cadeia1(i) - \sum_{k=1}^n cadeia2(k) * Menos5(i, k) \leq 0$$

#Garanto que no raio de 5km de um restaurante na cadeia2 haverá pelo menos

#um restaurante da cadeia1

$$cadeia2(i) - \sum_{k=1}^n cadeia1(k) * Menos5(i, k) \leq 0$$

Problema 4 -> Problema de transporte de combustíveis.

Formulação

Variáveis

- n – Lida do arquivo de entrada com o número de tanques do caminhão. (inteiro)
- m – Lida do arquivo de entrada com o número de tipos de combustíveis solicitados. (inteiro)
- $capacidade$ – Lida do arquivo de entrada com o array com as capacidades dos n tanques do caminhão. (real)
- $demanda$ – Array com a quantidade demandada dos m combustíveis; (inteiro)
- $quantoLevo$ – Matriz $n \times m$ de variáveis do modelo com a quantidade de combustível levada. (inteiro)
- $carregadoCombust$ – Matriz $n \times m$ de variáveis do modelo informando qual combustível é levado em determinado tanque. (binário)

Objetivo

$$\text{Min} \sum_{i=1}^m demanda(i) - \sum_{i=1}^n \sum_{j=1}^m quantoLevo(i, j)$$

Restrições

Para todo i de 1 até n faça

#Garanto que só levo um combustível por tanque.

$$\sum_{j=1}^m carregadoCombust(i, j) = 1$$

Fim-Para

Para todo j de 1 até m faça

#Garanto que só levo no máximo a quantidade demandada de combustível em um tanque.

$$\sum_{i=1}^n quantoLevo(i, j) \leq demanda(j)$$

Fim-Para

Para todo i de 1 até n faça

Para todo j de 1 até m faça

#Garanto que levo no máximo a capacidade do tanque.

$$quantoLevo(i, j) \leq capacidade(i) * carregadoCombust(i, j)$$

Fim-Para

Fim-Para

```
Para todo i de 1 até n faça
  Para todo j de 1 m faça
    #Garanto que não levo mais que a quantidade demandada de combustível.
    quantoLevo(i,j) ≤ demanda(j) * carregadoCombust(i,j)
  Fim-Para
Fim-Para
```