UNIVERSIDADE FEDERAL DO PARANÁ

MARIA TERESA KRAVETZ ANDRIOLI (GRR20171602)
JEAN GUILHERME CARRARO DA SILVA (GRR20176542)

OTIMIZAÇÃO (CI 1238): TRABALHO 1

CURITIBA
2021

1 INTRODUÇÃO

Esse trabalho tem como objetivo fazer a leitura de um conjunto de números que representam sedes de uma empresa de tecnologia e o custo da conexão ponto-a-ponto entre elas e gerar uma saída a ser usada pelo resolvedor *lp_solve* para buscar uma solução que minimize os custos e atenda a demanda da empresa.

2 PROBLEMA

Uma empresa possui diversas sedes em uma cidade e quer ligar essas sedes através de conexões ponto-a-ponto de modo que o custo seja o menor possível. Essencialmente é um [3] problema do caminho mínimo, ou seja, uma questão da teoria dos grafos que busca minimizar o custo total do caminho entre dois vértices.

3 IMPLEMENTAÇÃO

Para a implementação, primeiro é feita a leitura de todos os dados de entrada. Depois, é criado um vetor com o *id* de cada sede (*[1...num sedes]*). Com esse vetor e o vetor de custos lido na entrada, é feito um dicionário cujas chaves são uma tupla (*o*,*d*) nas quais *o* é a sede de origem e *d* é a sede de destino. O valor das chaves é o custo da transmissão entre essas sedes. Como os custos são nas duas direções, cada direção possui uma chave e um valor no dicionário. Por exemplo, se o custo entre a sede 1 e a 2 é 5, existe uma chave (1,2) = 5 e uma (2,1) = 5.

Depois de feita essa leitura dos dados, o que resta é apenas formatar o arquivo final (com nome *output.lp*) no formato do *lp_solve*. Para isso, existe a função *criaSaida()*, que recebe como parâmetro o vetor de sedes, o dicionário *custos_conexoes*, o número de conexões, a sede de origem e a sede de destino. Nessa função, é aplicado um algoritmo de criação de programas lineares para o problema do caminho mínimo em grafos de duas direções segundo a seção Shortest Path Problem do [4]. Nessa seção, explica-se que nesse caso, o programa busca achar o mínimo do somatório de todos vértices (ida e volta), cada um multiplicado pelos seus custos multiplicados pela demanda:

$$min \sum_{todos\ os\ vértices} demanda * custo_{ij} * x_{ij}$$

Além disso, aplica-se as seguintes fórmulas:

$$\sum_{vértices\ saindo\ de\ i} x_{ij} - \sum_{vértices\ chegando\ em\ i} x_{ij} = 1 \;\; Para\ a\ sede\ de\ origem$$

$$\sum_{vértices\ chegando\ em\ i} x_{ij} - \sum_{vértices\ saindo\ de\ i} x_{ij} = 1 \;\; Para\ a\ sede\ de\ destino$$

$$\sum_{vértices\ saindo\ de\ i} x_{ij} - \sum_{vértices\ chegando\ em\ i} x_{ij} = 0 \;\; Para\ as\ sedes\ intermediárias$$

4 EXEMPLO

Usando o seguinte exemplo da especificação (teste.in):

```
3 3
1 3 10
1 2 2
2 3 3
1 3 6
```

O programa gera o seguinte output.lp:

```
min: 20.0 x12 + 20.0 x21 + 30.0 x23 + 30.0 x32 + 60.0 x13 + 60.0 x31;
x12 + x13 + -x21 + -x31 = 1;
x23 + x13 + -x32 + -x31 = 1;
x21 + x23 + -x12 + -x32 = 0;
```

Que gera o seguinte resultado após ser usado como entrada do lp_solve:

```
Value of objective function: 50.00000000
Actual values of the variables:
x12                              1
x21                              0
x23                              1
x32                              0
x13                              0
x31                              0
```

5 EXECUÇÃO

make gera o executável *escolha*

./escolha < **arquivo_de_teste**

6 GITHUB

Link do repositório no github contendo todos os arquivos do projeto: https://github.com/mariaandrioli/otimizacao

REFERÊNCIAS

[1] "Python - Graphs". *Tutorialspoint.Com*, 2021, https://www.tutorialspoint.com/python_data_structure/python_graphs.htm.

[2] Matoušek, Jiří, and Birgit Gärtner. Understanding And Using Linear Programming. Springer, 2007.

[3] "Problema Do Caminho Mínimo – Wikipédia, A Enciclopédia Livre". Pt.Wikipedia.Org, 2021, https://pt.wikipedia.org/wiki/Problema_do_caminho_mínimo.

[4] Arsham, Hossein. "Integer Programs And Network Models". Home.Ubalt.Edu, 2021, http://home.ubalt.edu/ntsbarsh/opre640a/partIII.htm#rmaxflpr.