Universidade Federal de Lavras

Departamento de Ciência da Computação

TRABALHO PRÁTICO - PROGRAMAÇÃO MATEMÁTICA

Aluno:Adson Gregorio

João Paulo Costa

Jonhy Geraldo da Silva

Lucas Pancotto Zorzi

Vinícius Salles

AGOSTO/2016

Sumário

1. Introdução

O presente trabalho objetiva resolver um problema de roteamento de veículos da disciplina GCC-118 Programação Matemática, em súmula o trabalho consiste em: Dado um conjunto de elementos a serem transportados (no caso em questão conjunto de pessoas), uma quantidade de veículos para transporta-los, um conjunto de eventos definidos e valores definidos para o deslocamento de um evento para o outro. Para resolver esse problema deve-se criar uma função que objetiva minimizar o custo total do deslocamento de todos esses elementos.
Cada elemento possui um evento destino e um turno para ser transportado, podendo ser, Vespertino, Matutino e Noturno, o problema define que não se pode mapear elementos com turno Matutino e Noturno em um mesmo veículo.

2. Descrição

Considere os seguintes parâmetros de entrada:

- A: Conjunto de atividades escolhidas pelos diversos turistas, isto é, A = {a1, a2,..,an}
- C_{ij}: Custo de ligação entre as atividades i e j de A
- n: Quantidade de turistas
- Q: Custo fixo do carro

Variáveis de decisão:

- X_{ijk}: variável binária que assume valor 1 se o arco (i,j) for utilizado pelo carro k e 0, caso contrário
- Y_k: variável binária que assume valor 1 se o carro k for utilizado e 0, caso contráro

Para iniciar a modelagem do problema proposto utilizando a biblioteca PuLP. Foi criada uma variável chamada model que recebe o retorno da função LpProblem. Esta função possui dois parâmetros, o primeiro é uma string e representa o nome do problema e o segundo parâmetro, no nosso caso, é o LpMinimize pois temos como objetivo minimizar o custo da função objetivo.

```
model = pulp.LpProblem("PTTOR-16", LpMinimize)
```

2.1.Função objetivo

A função objetivo proposta tem o objetivo de minimizar o custo do transporte de turista.

$$MIN \sum_{k=1}^{n} \sum_{i,j=1}^{vertices} c_{ijk} * x_{ijk} + \sum_{k=1}^{n} y_k * Q$$

```
model += lpSum(lpSum(custos[i][j]*x[i][j][k] for i in escolhas for j in escolhas)
        for k in numeroCarros) + lpSum( y[k]*custoCarro for k in numeroCarros)
```

2.2. Restrições

A seguir são descritas as restrições utilizadas para resolução do problema.

Restrição 1. Restringe que exista trajetos diretamente ligados entre turnos matutino e noturno

```
for k in numeroCarros:
        for i in escolhas:
                for j in escolhas:
                        if(v[i] == 'N' and v[j] == 'M'):
                                model +=( x[i][j][k]) == 0
                        if(v[j] == 'M' and v[i] == 'N'):
                                model += (x[j][i][k]) == 0
```

Restrição 2. Restringe trajetos indiretamente ligados com turnos matutino e noturno

```
for k in numeroCarros:
        for i in escolhas:
                for j in escolhas:
                        for n in escolhas:
                                if(v[i] == 'N' and v[n] == 'V' and v[j] == 'M'):
                                        model += (x[i][n][k] + x[n][j][k]) <= 1
                                if(v[i] == 'M' and v[n] == 'V' and v[j] == 'N'):
                                        model += (x[i][n][k] + x[n][j][k]) <= 1
```

Restrição 3. Restringe que não seja criados sub-ciclos dentro de um percurso

$$\sum_{k=1}^{n} x_{ijk} \leq 1 \quad \forall i \, \varepsilon \, A, \forall j \, \varepsilon \, A$$

```
for i in escolhas:
        for j in escolhas:
                model +=  lpSum( x[j][i][k] + x[i][j][k]    for k in numeroCarros)  <= 1
```

Restrição 4. Restrição para garantir que cada vértice terá uma única saída. O único vértice que pode ter mais que uma aresta de saída é o vértice 0. O vértice 0(saída) possui um número minimo de arestas que é igual ao numero total de clientes dividido pela capacidade dos carros, arredondando esses valores para cima, no caso dos exemplos, o número minimo foi 3.

$$\sum_{i=1}^{vertices} \sum_{k=1}^{n} x_{ijk} = 1 \ \forall j \neq 0$$

ou

$$\sum_{i=1}^{vertices} \sum_{k=1}^{carros} x_{ijk} \geq 3 \ \forall j = 0$$

```
for j in escolhas:
        if(j != 0):
                model += lpSum( x[j][i][k]  for i in escolhas for k in numeroCarros) == 1
        else:
                #restringe um numero minimo de carros saindo do ponto 0
                model += lpSum( x[j][i][k]  for i in escolhas for k in numeroCarros) >= 3
```

Restrição 5. Restringe o número de entrada no vértice, cada vértice possui apenas uma aresta de entrada. O único vértice que pode ter mais de uma aresta chegando a ele é o vértice 0 (mesmo caso da saída).

$$\sum_{j=1}^{vertices} \sum_{k=1}^{n} x_{ijk} = 1 \ \forall i \neq 0$$

$$\sum_{j=1}^{vertices} \sum_{k=1}^{n} x_{ijk} \geq 3 \ \forall i = 0$$

```
for i in escolhas:
        if(i != 0):
                model += lpSum( x[j][i][k]  for j in escolhas for k in numeroCarros) == 1
        else:
                #restringe um numero minimo de carros chegando no ponto 0
                model += lpSum( x[j][i][k]  for j in escolhas for k in numeroCarros) >= 3
```

Restrição 6. Garante que se um carro chegue a um vértice, o mesmo carro terá que sair desse vértice.

$$\sum_{k=1}^{n} x_{jik} - x_{ijk} = 0 \ \forall i \ \varepsilon \ A, \forall j \ \varepsilon \ A$$

```
for k in numeroCarros:
        for i in escolhas:
                model +=  lpSum( x[j][i][k] - x[i][j][k]    for j in escolhas  )  == 0
```

Restrição 7. Restrição para que existam no máximo 4 passageiros no carro, assim o somatório das arestas que o carro percorrer nunca sera maior que 5,

$$\sum_{j=1}^{vertices} \sum_{i=1}^{carros} x_{jik} \leq 5 \quad k = \{1,...,n\}$$

```
for k in numeroCarros:
        model += lpSum( x[j][i][k]  for j in escolhas for i in escolhas) <= 5
```

Restrição 8. Restrição para avaliar se o carro será utilizado, se sim a variável y terá valor 1, caso contrario 0, essa variável é utilizada na função objetiva para calcular o custo fixo .

$$y_k \geq x_{jik} \quad \forall i \, \varepsilon \, A, \forall j \, \varepsilon \, A, k = \{1,...,n\}$$

```
for k in numeroCarros:
        for j in escolhas:
                for i in escolhas:
                        model += y[k] >= x[j][i][k]
```

Restrição 9. Integralidade e não-negatividade

$$x_{ijk} \, \varepsilon \, \{1,0\} \quad \forall i \, \varepsilon \, A, \forall j \, \varepsilon \, A, k = \{1,...,n\}$$

$$y_k \, \varepsilon \, \{1,0\} \quad k = \{1,...,n\}$$

$$c_{ijk} \geq 0 \quad \forall i \, \varepsilon \, A, \forall j \, \varepsilon \, A, k = \{1,...,n\}$$

3.Experimentos computacionais

Para resolução do problema foi utilizado a linguagem python na versão 2.7.11 juntamente com a biblioteca PuLP na versão 1.6. A utilização desta biblioteca proporcionou uma maneira eficiente para alcançar a otimização do problema tratado.

Para a resolução das instâncias em um tempo viável, adotamos a estratégia de dividir a execução entre todos os integrantes do grupo. Assim sendo, cada integrante do grupo executou em média 8 instâncias. Cada computador utilizado apresenta as seguintes configurações:

Computador 1

Sistema Operacional: Windows 10 Home Single Language

Processador: i5-6200U CPU @ 2.30GHz 2.4GHz

Memória Instalada (RAM): 4GB

Tipo de sistema: Sistema Operacional 64 bits, processador com base em x64

Computador 2
Sistema Operacional: Linux Ubuntu 16.04
Processador: Intel(R) Core(TM) i3-3227U CPU @ 1.90GHz
Memória Instalada(RAM): 4GB
Tipo de Sistema: Sistema Operacional de 64 bits, processador com base x64

Computador 3
Sistema Operacional: Windows 10 Home Single Language
Processador: Intel Core i7-4510U CPU @ 2,0GHz 2,6GHz
Memória Instalada (RAM): 8,00GB
Tipo de Sistema: Sistema Operacional de 64 bits, processador com base x64

Computador 4
Sistema Operacional: Windows 10 Home Single Language
Processador: Intel Core i7-5500U CPU @ 2,4GHz 2,4GHz
Memória Instalada (RAM): 4,00GB
Tipo de Sistema: Sistema Operacional de 64 bits, processador com base x64

Computador 5
Sistema Operacional: Windows 8.1
Processador: Intel Core i7-4510U CPU @ 2.00GNz 2.60 GHz
Memória Instalada (RAM): 8,00GB
Tipo de Sistema: Sistema Operacional de 64 bits, processador com base x64

3.1.Resultados

3.1.1. Resultados obtidos para a instância Rio16.txt

- **Solução Ótima (Custo final):** 1073
- **Número de Carros:** 3
- **Turistas no carro 6:**
  - 42, 36, 44
- **Turistas do carro 7:**
  - 13, 28, 27, 37
- **Turistas do carro 10:**
  - 12, 50, 43
- **Trajeto do carro 6:**
  - Saída → 42 → 36 → 44 → Chegada
- **Trajeto do carro 7:**

- Saída → 13 → 28 → 37 → 27 → Chegada

- **Trajeto do carro 10:**
  - Saída → 12 → 50 → 43 → Chegada

3.1.2.Resultado obtido para as 40 instâncias PTTOR-16

Instância	Status	Computador Utilizado	GAP	Custo total	Numero carros	Tempo (s)
I1.PTTOR-16	Not Solved	Comp. 1	0.05	1013	3	3600.27
I2.PTTOR-16	Not Solved	Comp. 1	0.14	987.00	3	3600.0
I3.PTTOR-16	Not Solved	Comp. 1	0.15	883.00	3	3600.0
I4.PTTOR-16	Optimal	Comp. 1	-	952	3	35.20
I5.PTTOR-16	Optimal	Comp. 1	-	938.00	3	418.95
I6.PTTOR-16	Not Solved	Comp. 1	0.06	985.00	3	3600.39
I7.PTTOR-16	Optimal	Comp. 1	-	1086.00	3	1702.70
I8.PTTOR-16	Optimal	Comp. 1	-	913.00	3	2480.21
I9.PTTOR-16	Optimal	Comp. 2	-	978.00	3	79.5
I10.PTTOR-16	Optimal	Comp. 2	-	1085.00	3	13.11
I11.PTTOR-16	Optimal	Comp. 2	-	896.00	3	3.44
I12.PTTOR-16	Optimal	Comp. 2	-	1207.00	3	455.37
I13.PTTOR-16	Not Solved	Comp. 2	0.16	1124.00	3	3599.34
I14.PTTOR-16	Not Solved	Comp. 2	0.08	1028.00	3	3599.90
I15.PTTOR-16	Optimal	Comp. 5	-	964.00	3	1487.19
I16.PTTOR-16	Optimal	Comp. 5	-	1022.00	3	28.26
I17.PTTOR-16	Optimal	Comp. 3	-	1102.00	3	2.57
I18.PTTOR-16	Optimal	Comp. 3	-	958.00	3	1097.43
I19.PTTOR-16	Not Solved	Comp. 3	0.11	926.00	3	3599.56

I20.PTTOR-16	Optimal	Comp. 3	-	985.00	3	85.50
I21.PTTOR-16	Optimal	Comp. 3	-	703.00	3	16.35
I22.PTTOR-16	Not Solved	Comp. 3	0,02	731.00	3	3599.22
I23.PTTOR-16	Not Solved	Comp. 3	0.03	554.00	3	3599.47
I24.PTTOR-16	Optimal	Comp. 3	-	636.00	3	2755.86
I25.PTTOR-16	Optimal	Comp. 4	-	616.00	3	1095.45
I26.PTTOR-16	Optimal	Comp. 4	-	804.00	3	485.86
I27.PTTOR-16	Optimal	Comp. 4	-	598.00	3	49.78
I28.PTTOR-16	Optimal	Comp. 4	-	706.00	3	291.07
I29.PTTOR-16	Optimal	Comp. 4	-	658.00	3	612.38
I30.PTTOR-16	Optimal	Comp. 4	-	525.00	3	95.79
I31.PTTOR-16	Optimal	Comp. 3	-	725.00	3	857.59
I32.PTTOR-16	Optimal	Comp. 4	-	0650.0	3	17.42
I33.PTTOR-16	Not Solved	Comp. 5	0.17	679.00	3	3599.67
I34.PTTOR-16	Not Solved	Comp. 3	0.03	638.00	3	3599.07
I35.PTTOR-16	Optimal	Comp. 5	-	605.00	3	1839.47
I36.PTTOR-16	Optimal	Comp. 5	-	587.00	3	29.85
I37.PTTOR-16	Optimal	Comp. 5	-	603.00	3	1795.02
I38.PTTOR-16	Not Solved	Comp. 5	0.05	612	3	3599.59
I39.PTTOR-16	Optimal	Comp. 5	_	609	3	284.78
I40.PTTOR-16	Optimal	Comp. 5	-	615.00	3	226.85

3.1.3. Avaliação das soluções

Soluções ótimas: 28 soluções ótimas encontradas

Somatório dos tempos de execução: 44482,52 segundos
Tempo médio de execução: 1112,063 segundos
Menor GAP: 0,02
Maior GAP: 0.25

Uma lista com os resultados detalhados de todas as instâncias pode ser encontrada no arquivo "Anexo1".

4. Comentários finais

Sobre o desenvolvimento do trabalho, o grupo considera que, a utilização do Python foi um empecilho menor do que a formulação matemática das restrições, isso pois, os membros do grupo estavam habituados a definir restrições utilizando comandos tais qual "if" e "else", comandos que para o grupo são bem menos complexos e portanto mais fáceis de serem implementados. A dificuldade encontrada na utilização do Python, estava na compreensão da biblioteca PuLP, em que seria necessário, por exemplo, utilizar a função LPsum, para mapear o somatório do modelo matemático.
Apesar de todas as dificuldades encontradas, o grupo acredita ter concluído o trabalho de forma satisfatória, tendo em vista que o resultado, da instância Rio16.txt, passado para conferencia, confere com o resultado obtido via nosso modelo.

5. Referências

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. Pesquisa operacional para cursos de engenharia. Editora Campus, 2007.

Jamilson F. Souza, Marcone. Otimização Combinatória, Notas de aula, 2009/2
Departamento de Computação, Universidade Federal de Ouro Preto.

Optimization with PuLP - <https://pythonhosted.org/PuLP/>