UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

OTIMIZAÇÃO DE SISTEMAS

ANÁLISE DE UM PROBLEMA E MODELAGEM VIA PROGRAMAÇÃO LINEAR INTEIRA MISTA (PLIM).

Alunos: Liza Macedo Lopes e Reginaldo Ferreira

1 Descrição detalhada do problema

O presente projeto visa determinar a rota de menor custo que conecte um número **n** de cidades que se pretende visitar. Como estudo de caso, imaginemos o seguinte problema: Pretende-se fazer uma viagem pela Europa e espera-se visitar 10 cidades. Para cada cidade definimos uma quantidade mínima e máxima de dias de estadia. A cidade de chegada na Europa será Lisboa, no dia 01/01/2020. A data de retorno para o Brasil será no dia 30/01/2020, saindo também de Lisboa. O objetivo então será determinar a rota de menor custo total respeitando os tempos de estadia mínima e máxima em cada cidade. O custo das passagens de ida e volta até Lisboa não será considerado no problema. Supondo que os preços das passagens aéreas possam ser considerados aproximadamente proporcional à distância entre as mesmas, o problema pode ser considerado como uma variação do problema do caixeiro viajante. Porém, neste caso, diferentemente do problema do caixeiro viajante clássico, teremos que o considerar também os tempos de estadia em cada cidade.

Para este caso, os tempos de viagem entre as cidade foram desprezados pois consideramos que os voos entre as cidade na europa são de curta duração quando comparados aos prazos de estadia mínima em cada cidade.

Antes de prosseguir, é importante definirmos a nomenclatura e as abreviações que serão utilizadas no texto.

Tabela 1 – abreviações que serão utilizadas no texto

Termo	Abreviação	Definição
Cidade de chegada	C_0	cidade por onde se inicia e onde termina a rota. Cidade de chegada na Europa
Trecho	x_{ij}	trecho ligando a cidade i a cidade j, sendo i a cidade de origem e j a a cidade de destino. O indize zero refere-se à cidade de chegada .
Custo	c_{ij}	custo associado a trecho x_{ij}
Rota	-	conjunto dos trechos escolhidos
dias de viagem	ddv	número de dias que se pretende viajar
voos por dia	vpd	quantidade de voos que serão considerados a cada dia.

2 Descrição do modelo

Dado um conjunto de **n** cidades, o número de possíveis trechos ligando-as entre si é dado pela combinação:

$$C(n,2) = \frac{n!}{(n-2)!} \tag{1}$$

No caso em análise, para cada trecho possível, será considerada a existência de um número fixo de voos por dia (vpd) para cada um dos dias de viagem (ddv). Por exemplo, supondo que

o número de cidades que gostaríamos de visitar seja igual a 4, o tempo de viagem seja igual a 15 dias e que os voos considerados sejam os 5 mais baratos em cada um dos dias. Neste caso, o número total de passagens (trechos) será igual a 900. Ou seja, $n \cdot (n-1) \cdot vpd \cdot ddv$.

A função objetivo que deve ser minimizada pode ser escrita da seguinte forma:

$$min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} x_{ij} \tag{2}$$

Onde

$$x_{ij} = \begin{cases} 1, & \text{em caso de uso trecho entre as cidades i e j} \\ 0, & \text{caso contrário} \end{cases} \tag{3}$$

e c_{ij} representa o preço da passagem associada a cada trecho. Considerando que cada cidade será visitada uma única vez, temos as seguintes restrições:

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1 \quad \forall i \in \aleph \tag{4}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1 \quad \forall j \in \aleph \tag{5}$$

Onde $\aleph$ é o conjunto formados pelas n cidades que se pretende visitar. As restrições 4 e 5 não excluem a possibilidade de que a solução encontrada apresente subrotas. Ou seja, não impede que o algoritimo encontre soluções com rotas conectando apenas um subconjunto do total de cidades (Figura 1).

Na formulação apresentada neste trabalho, no entanto, soluções com subrotas são excluídas por conta da adoção das restrições de tempo. A restrição de tempo impõe que a saída de uma determinada cidade deve ocorrer após a chegada nesta mesma cidade mais o tempo mínimo de estadia. E é também necessário que esta saída ocorra antes a chegada mais o tempo máximo de estadia. Ou seja:

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} \Delta T_{ij} > \sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ji} \Delta T_{ji} + emin_i \quad \forall i, j \in \aleph \tag{6}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} \Delta T_{ij} < \sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ji} \Delta T_{ji} + emax_i \quad \forall i, j \in \aleph \tag{7}$$

Onde ΔT_{ij} é o tempo total (em segundos) de viagem em relação ao momento de chegada em C_0. Por exemplo, supondo que a chegada em Lisboa ocorra no dia 01/01/2020 às 00:00 e

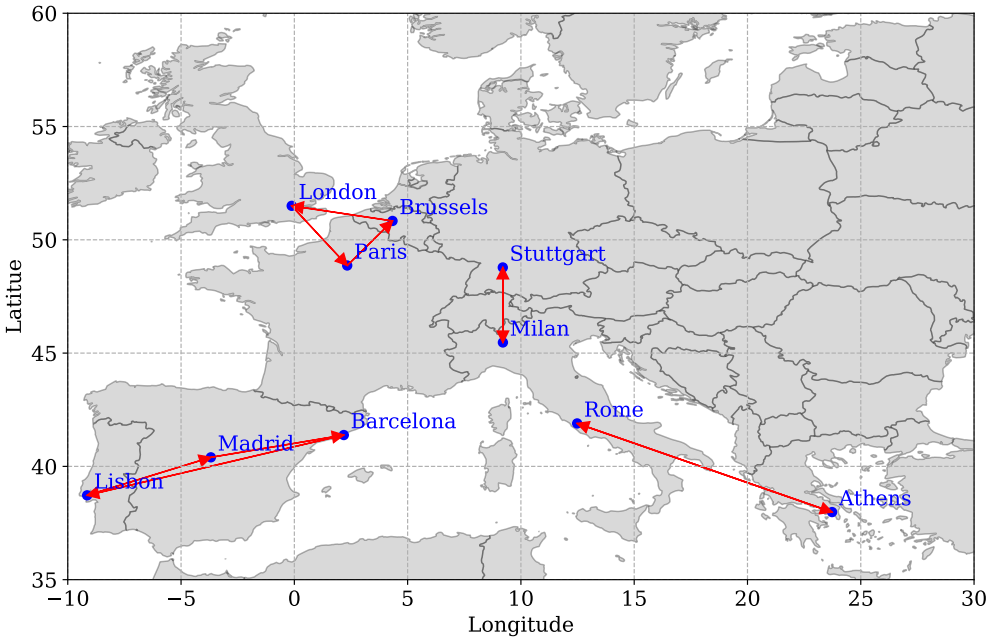

Figura 1 – Representação de algumas subrotas possíveis

que a saída de Lisboa seja para Paris no dia 03/01/2020 às 13h00. Então $\Delta T_{Lisboa-Paris}$ será igual a 2 dias e 13 horas ou 219600 segundos. A restrições 6 e 7 aplicam-se a todas as cidades que se pretende visitar exceto para a C_0, já que iniciaremos o tour a partir dessa cidade e, ao final, retornaremos à ela. Consequentemente, nesse caso, a saída ocorrerá antes da chegada. Para C_0 a restrição dos tempos deve considerar apenas as estadias mínimas e máximas:

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} \Delta T_{ij} > emin_i \quad \forall i, j \in \aleph \tag{8}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} \Delta T_{ij} < emax_i \quad \forall i, j \in \aleph \tag{9}$$

Os dados de entrada utilizados no modelo foram gerados artificialmente. Inicialmente foi tentado uma busca de dados reais dos preços de passagens através da utilização do API Skyscanner, mas devido a dificuldade de implementação do código e, principalmente, ao baixo desempenho do tempo de resposta da versão gratuita, optamos por criar um dataset artificial. Os detalhes dos parâmetros para a geração do dataset são apresentados em seguida.

3 Implementação computacional

Inicialmente, foi necessário criar um dataset contendo voos com diferentes preços e horários de passagens. Como cada voo está associado a um trecho, o número total de voos é dado por $n \cdot (n-1) \cdot vpd \cdot ddv$. O preço médio das passagens entre as cidades é proporcional à distância entre as mesmas. Porém, para que estes preços não fossem exatamente iguais entre duas determinadas cidades, o valor final é o preço médio somado a um determinado valor aleatório também proporcional à média. O preço médio das passagens foi obtido considerando o preço por km (ppkm) igual a US$ 0,10; o parâmetro σ adotado foi de 0,3. A figura 2 ilustra como o custo final da passagem é calculado.

Quanto ao número de dias de viagem, deve-se cuidar apenas para que este seja maior que a soma das estadias mínimas e cada cidade e menor que a soma das estadias máximas. No nosso exemplo consideramos uma viagem de 30 dias.

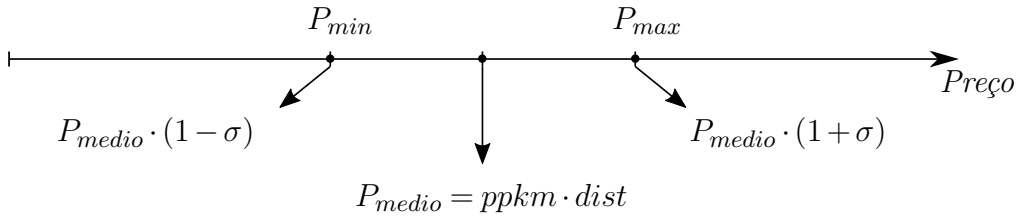

Figura 2 – Preço das passagens: o valor é uma escolha aleatória entre P_{min} e P_{max}

Para a obtenção das distâncias entre as cidades, foi criada uma função que retorna as coordenadas de latitude e longitude das **n** cidades. Os dados de coordenadas de cada cidade são retirados de uma lista com mais e 12000 cidades. Os dados são então utilizados por outra função para que esta retorne as distâncias entre as cidades. Toda a implementação computacional foi feita em Python e o código completo pode ser acessado e executado na versão colab do Google através do seguinte link: https://colab.research.google.com/drive/1Z3v8M1DV_ZIFXFaBfusZzR8S8pqBKjcf.

3.1 Implementação da função objetivo:

O Solver utilizado para a solução do problema foi o Pulp que roda em Python. A implementação da função objetivo (2) foi feito da seguinte forma:

Listing 1: Função objetivo

```
# pVars: Variaveis binarias assiciadas ao trechos
pVars = pl.LpVariable.dicts('', 0, 1, pl.LpBinary)

# set_of_pulp_vars: Array com as variaveis binarias (pVars)
set_of_pulp_vars = np.array(list(pVars.values()))

# precos: Array com os precos de cada trecho
# fobj: Funcao objetivo

fobj =  set_of_pulp_vars * precos

# prob: variavel Pulp do tipo LpProblem
prob = pl.LpProblem('', pl.LpMinimize)
prob += pl.lpSum(fobj)
```

3.2 Implementação das restrições do modelo:

Listing 2: Implementação das restrições 1 e 2

```
# dpv_df: dataframe com os precos dos voos, horarios,
#         cidade de origem e destino
# ListCidade: lisca com o nome das cidades
listCidades = pdv_df['cityName'].to_numpy()
for i in listCidades:
    mask = (origem != i) & (destino == i)
    index = trechos_np[mask][:,0].astype(int)
    prob += pl.lpSum(set_of_pulp_vars[index]) == 1

for i in listCidades:
    mask = (origem == i) & (destino != i)
    index = trechos_np[mask][:,0].astype(int)
    prob += pl.lpSum(set_of_pulp_vars[index]) == 1
```

Listing 3: Implementação das restrições 3 e 4

```
for i in listCidades[listCidades != cidade_de_chegada]:
    mask_Origem  = (trechos_np[:,1] == i)
    mask_Destino = (trechos_np[:,2] == i)
    emin = float(pdv_df[pdv_df.cityName == i].eMin) * 24 * 3600
    emax = float(pdv_df[pdv_df.cityName == i].eMax) * 24 * 3600

    a = set_of_pulp_vars[mask_Origem]
    b = trechos_np[mask_Origem][:,-1].astype(float)
    c = set_of_pulp_vars[mask_Destino]
    d = trechos_np[mask_Destino][:,-1].astype(float)

    possiveis_saidas   = np.sum(b * a )
    possivais_chegadas = np.sum(c * d )

    prob += (possiveis_saidas - possivais_chegadas - emin) >= 0
    prob += (possiveis_saidas - possivais_chegadas - emax) <= 0
```

Listing 4: Implementação das restrições 5 e 6

```
 for i in listCidades[listCidades != cidade_de_chegada]:
    mask_Origem  = (trechos_np[:,1] == cidade_de_chegada)
    mask_Destino = (trechos_np[:,2] == i)
    emin = float(pdv_df[pdv_df.cityName ==
                        cidade_de_chegada].eMin) * 24 * 3600
    emax = float(pdv_df[pdv_df.cityName ==
                        cidade_de_chegada].eMax) * 24 * 3600

    a = set_of_pulp_vars[mask_Origem]
    b = trechos_np[mask_Origem][:,-1].astype(float)
    possiveis_saidas   = (b * a )

    prob += pl.lpSum(possiveis_saidas) >= emin
    prob += pl.lpSum(possiveis_saidas) <= emax
```

3.2.1 Rodando o modelo

Listing 5: Rodando o modelo

```
prob.solve()
```

4 Resultados Obtidos

Como dito anteriormente, como exemplo de estudo, consideramos uma viagem para a Europa com visitas em 10 cidades. O nome de cada cidade e os tempos de estadia mínima e máxima são apresentados na tabela 2. Em nosso plano de viagem, a cidade de chegada será Lisboa e o tempo total de viagem será de 30 dias. Os preços considerados são, para cada um dos trechos, os 5 mais baratos de cada dia. Temos então que o total de variáveis binárias do nosso modelo será igual a 13500. A dez primeiras linhas do nosso dataset são apresentadas na tabela 3.

Cidade	País	eMin	eMax
Stuttgart	Germany	2	4
Lisbon	Portugal	2	5
Brussels	Belgium	3	6
Milan	Italy	2	4
Paris	France	2	4
Madrid	Spain	2	4
Barcelona	Spain	5	7
Rome	Italy	2	4
Athens	Greece	3	6
London	United Ki	2	4

Tabela 2 – Plano de viagem

O tempo de processamento do solver foi de aproximadamente 3.5 minutos e os resultados estão apresentados na tabela 4. O valor da função objetivo foi de 538.2599. Ou seja, o custo com passagens para realizarmos o tour pela cidades escolhidas seria de US$ 538.26. A figura 3 ilustra a rota escolhida.

Origem	Destino	Distancia (km)	Preço (US$)	Horários-Voos
Milan	Paris	641.85	68.30	2020-01-02 00:05:00
Brussels	Madrid	1315.25	124.47	2020-01-02 00:05:00
Milan	Brussels	696.98	86.43	2020-01-02 00:11:00
Stuttgart	Barcelona	988.98	114.45	2020-01-02 00:11:00
Stuttgart	Brussels	417.12	49.47	2020-01-02 00:11:00
Milan	Lisbon	1684.40	202.21	2020-01-02 00:21:00
Rome	Paris	1106.89	134.13	2020-01-02 00:22:00
Paris	Stuttgart	502.62	42.07	2020-01-02 00:22:00
London	Milan	958.59	70.36	2020-01-02 00:23:00
London	Paris	340.79	23.98	2020-01-02 00:24:00

Tabela 3 – Dez primeiras linhas do dataset gerado. As linhas estão ordenadas por Horários-Voos

Origem	Destino	Horários-Voos	V. Total (US$)	Estadia na Origem
Lisbon	London	2020-01-03 04:03:00	113.17	2 days 04:03:00
London	Paris	2020-01-05 18:24:00	28.14	2 days 14:21:00
Paris	Brussels	2020-01-07 21:24:00	20.44	2 days 03:00:00
Brussels	Stuttgart	2020-01-10 22:32:00	29.35	3 days 01:08:00
Stuttgart	Milan	2020-01-13 12:23:00	26.72	2 days 13:51:00
Milan	Athens	2020-01-16 05:39:00	109.13	2 days 17:16:00
Athens	Rome	2020-01-20 16:49:00	75.57	4 days 11:10:00
Rome	Barcelona	2020-01-23 12:40:00	60.40	2 days 19:51:00
Barcelona	Madrid	2020-01-28 17:55:00	38.41	5 days 05:15:00
Madrid	Lisbon	2020-01-31 16:25:00	36.93	2 days 22:30:00

Tabela 4 – Dez primeiras linhas do dataset gerado. As linhas estão ordenadas por Horários-Voos

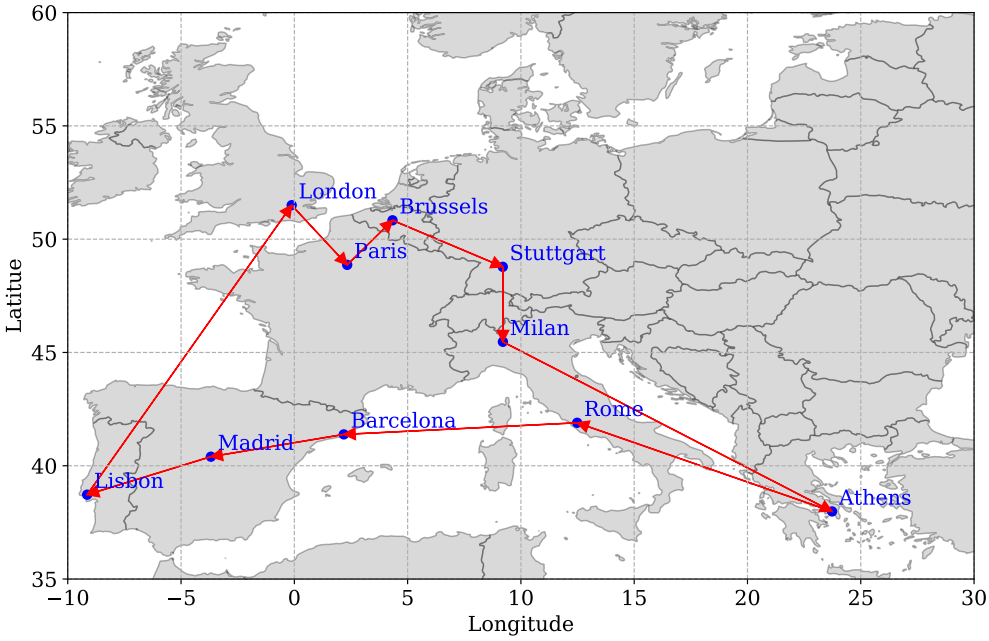

Figura 3 – Rota com menor preço entre as cidade selecionadas