



Um Enigma das Galáxias - Parte 1

SME0110 - Programação Matemática

Julia (Nº USP XXXXX) - email

Michelle (Nº USP XXXXX) - email

Matheus Carvalho Raimundo (Nº 10369014) - mcarvalhor@usp.br

Marcelo Duchêne (Nº 8596351) - marcelo.duchene@usp.br

Prof. Franklina Toledo e Prof. Marina Andretta

USP - São Carlos

Novembro/2020

Índice Analítico

1. Modelagem	3
1.1 Variáveis	4
1.2 Função-objetivo	5
1.3 Restrições	6
1.4 Resultado	7
2. Toy-problem	8
2.1 Guia de Execução	9
2.2 Solução	10
3. Referências Bibliográficas	11

1. Modelagem

Iniciaremos nossa modelagem analisando o problema. O astrônomo deve mover o telescópio o mínimo possível, passando por todas as galáxias, e deseja descobrir qual caminho deve fazer para isso.

Esse problema se assemelha muito ao Problema do Caixeiro Viajante, e a modelagem lembra tal.

1.1 Variáveis

Vamos denotar em nossa modelagem como L o conjunto das galáxias (coordenadas x e y das galáxias). Ou seja, L_1 é o (x, y) da primeira galáxia, L_2 é o (x, y) da segunda, e assim por diante. Esse conjunto L é conhecido ao programa, pois é um dado lido e armazenado no início da execução. Define-se, então, como N o número de galáxias em L :

$$N = |L|$$

Além disso, definimos a matriz C como o custo, ou seja, C_{ij} guarda a distância que o telescópio leva para se locomover da galáxia i para a galáxia j . Esse conjunto C também é conhecido ao programa, pois é um dado calculado usando a função de Distância Euclidiana^[1]:

$$C_{ij} = \sqrt{(L_j.x - L_i.x)^2 + (L_j.y - L_i.y)^2}$$

Agora partimos para a variável que o programa deve descobrir como solução: o caminho. Vamos descrever esse caminho no formato de uma matriz Z . Basicamente Z_{ij} indica se, estando na galáxia L_i , deve-se mover em seguida para a galáxia L_j (1) ou não (0). Ou seja, para cada linha de Z , apenas uma coluna adota o valor 1, enquanto que todas as outras colunas ficam como 0 - indicando o caminho adequado a ser seguido. Tem-se então:

$$Z_{ij} = \begin{cases} 1, & \text{se vai de } L_i \text{ para } L_j \\ 0, & \text{caso contrário} \end{cases}$$

A matriz Z é desconhecida no começo da execução do algoritmo. Ao fim da execução, ela estará preenchida. Mais semanticamente, com essa variável pode-se traçar um caminho que o astrônomo deve seguir de maneira a minimizar os movimentos no telescópio.

1.2 Função-objetivo

A função-objetivo é bem simples: minimizar os movimentos do telescópio. Para isso, é só somar os caminhos andados, multiplicado pela distância percorrida. Dessa forma, se um caminho não é tomado (0), ele não é somado. Mas se um caminho é tomado (1), a distância dele é somada. Como queremos minimizar essa distância somada, na modelagem tem-se:

$$\min : \sum_{i=1}^N \sum_{j=1}^N C_{ij} \times Z_{ij}$$

Perceba que o vetor C é usado porque já se possui ele calculado no programa, enquanto que o vetor Z é o que será descoberto como solução.

1.3 Restrições

Primeiro, vamos definir que não é possível sair e entrar na mesma galáxia:

$$Z_{ii} = 0 \quad , \forall i \in \{1, 2, \dots, N\}$$

Agora definimos que, a partir de cada galáxia, deve-se existir exatamente um caminho possível que sai dela:

$$\sum_{j=1}^N Z_{ij} = 1 \quad , \forall i \in \{1, 2, \dots, N\}$$

Analogamente, definimos que partindo para qualquer galáxia, deve-se existir exatamente um caminho possível que chegue até ela:

$$\sum_{i=1}^N Z_{ij} = 1 \quad , \forall j \in \{1, 2, \dots, N\}$$

Agora precisamos eliminar os **subciclos ilegais**^[2]. Para isso usaremos a mesma formulação de Dantzig–Fulkerson–Johnson para o Problema do Caixeiro Viajante^[3]. Essa formulação consiste em analisar cada subconjunto S de L possível, ficando:

$$\sum_{i \in S} \sum_{j \neq i \in S} Z_{ij} \leq |S| - 1 \quad , \forall S \subseteq \{1, 2, \dots, N\}, |S| \geq 2$$

Note que cada subconjunto S denota uma restrição diferente, e tais são gerados iterativamente pelo programa - são conhecidos.

Por fim, devemos definir o domínio das nossas variáveis:

$$Z_{ij} \in N \quad , \forall i, j \in \{1, 2, \dots, N\}$$

No programa, é possível notar que as restrições de domínio são adicionadas durante a própria alocação das variáveis. Não é necessário definir o domínio das variáveis C e L porque são variáveis cujos valores já são fixos e conhecidos.

1.4 Resultado

Juntando tudo o que foi descrito acima, tem-se o modelo:

$$\min : \sum_{i=1}^N \sum_{j=1}^N C_{ij} \times Z_{ij}$$

Sujeito a :

$$Z_{ii} = 0, \quad \forall i \in \{1, 2, \dots, N\}$$

$$\sum_{j=1}^N Z_{ij} = 1, \quad \forall i \in \{1, 2, \dots, N\}$$

$$\sum_{i=1}^N Z_{ij} = 1, \quad \forall j \in \{1, 2, \dots, N\}$$

$$\sum_{i \in S} \sum_{j \neq i \in S} Z_{ij} \leq |S| - 1, \quad \forall S \subseteq \{1, 2, \dots, N\}, |S| \geq 2$$

$$Z_{ij} \in \{0, 1\}, \quad \forall i, j \in \{1, 2, \dots, N\}$$

2. Toy-problem

O programa foi testado com diversos casos de teste, mas determinamos um problema-exemplo (Toy-problem). O problema-exemplo escolhido se trata de localizações reais de pontos no universo observável, com dados obtidos do website Google Sky^[4]. Veja:



Imagem 1: imagem do website Google Sky^[4].

Definimos então nosso conjunto L como:

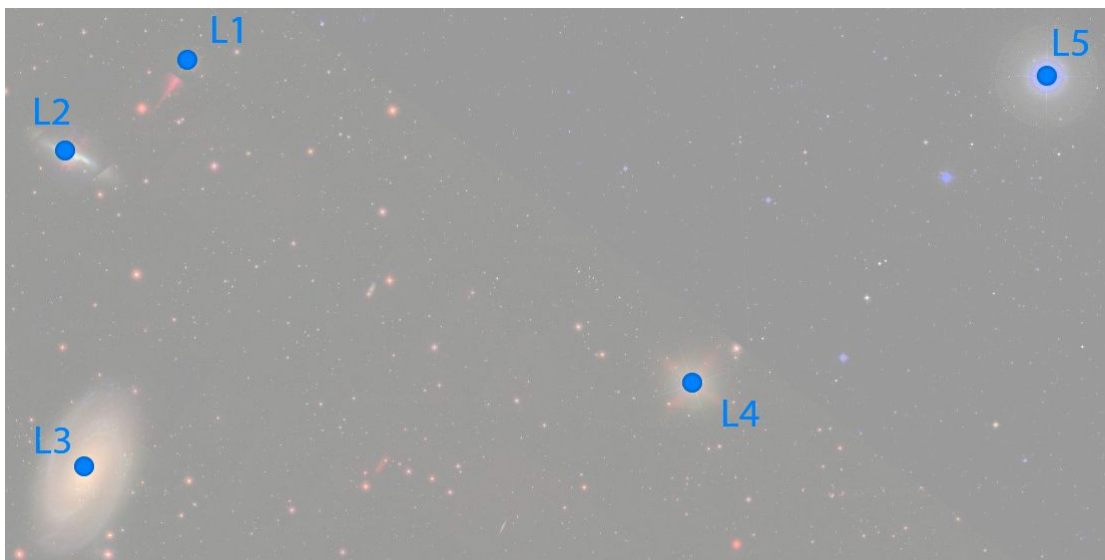


Imagem 2: conjunto L a partir de imagem do website Google Sky.

Por fim, temos as seguintes coordenadas (medida: segundos de arco^[5]):

L	x	y
1	35598	251501
2	35756	250852
3	35734	248644
4	34936	249270
5	34470	251399

Tabela 1: coordenadas do conjunto L .

O arquivo de entrada fornecido já está em formato textual que o programa consegue ler e reconhecer.

2.1 Guia de Execução

Para executar o programa, primeiramente é necessário instalar o ambiente de execução da linguagem Python 3, além de todas as bibliotecas necessárias - foi usada a biblioteca OR-Tools^[6]. Isso só é feito uma única vez antes da primeira execução. Execute no terminal:

```
> sudo apt-get install python3 python3-pip  
> sudo pip3 install -r requirements.txt
```

Para iniciar o programa é muito simples. Em um terminal, execute:

```
> python3 Solucao.py
```

O programa vai solicitar como entrada os dados para gerar os conjuntos L e C . Se for conveniente, é possível fornecer a entrada a partir de um arquivo. Dessa maneira não é necessário digitar os dados necessários toda vez que for executar o programa. Por exemplo, para fornecer como entrada o problema-exemplo do trabalho, em um terminal execute:

```
> python3 Solucao.py < CasosDeTeste/Toy-problem.tsp
```

2.2 Solução

Ao executar o programa, da maneira descrita na seção anterior para o problema-exemplo escolhido, obtém-se como resposta:

```
== Solução ==  
Valor da função objetivo: 7202.309  
Da galáxia 1, vamos para a galáxia 5.  
Da galáxia 2, vamos para a galáxia 1.  
Da galáxia 3, vamos para a galáxia 2.  
Da galáxia 4, vamos para a galáxia 3.  
Da galáxia 5, vamos para a galáxia 4.  
Caminho: 1 -> 5 -> 4 -> 3 -> 2 -> 1
```

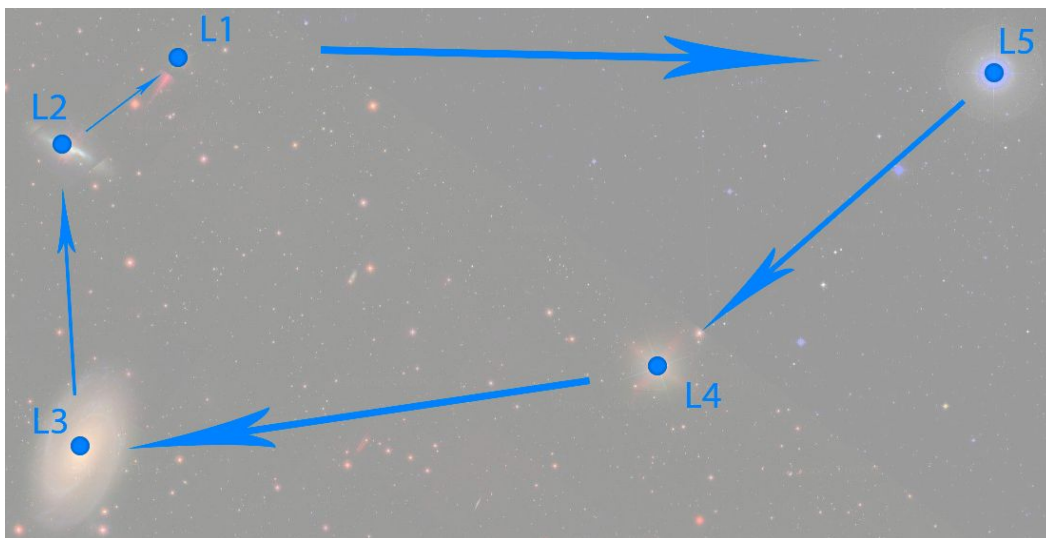


Imagem 3: representação gráfica da resposta do programa.

É evidente que tomar o caminho exatamente contrário a esse (caminhar na direção oposta) também é uma solução-ótima.

3. Referências Bibliográficas

1. “Distância Euclidiana”, disponível no website <https://en.wikipedia.org/wiki/Euclidean_distance> em 25 de outubro de 2020.
2. “Variáveis inteiras como ferramenta de modelagem”, por Prof. Franklina Toledo (aula do dia 09/09/2020), disponível no website <<https://edisciplinas.usp.br/>> em 25 de outubro de 2020.
3. “The Travelling Salesman Problem”, disponível no website <https://en.wikipedia.org/wiki/Travelling_salesman_problem> em 25 de outubro de 2020.
4. Google Sky, disponível no website <<https://www.google.com/sky/#latitude=69.27276405942452&longitude=-32.94125194551579&zoom=8>> em 30 de outubro de 2020.
5. “Conhecimento: Aprenda a estimar as distâncias que envolvem os objetos no céu”, disponível no website <https://www.apolo11.com/distancias_no_ceu.php> em 30 de outubro de 2020.
6. “OR-Tools”, disponível no website <<https://developers.google.com/optimization>> em 30 de outubro de 2020.