Relatório
Programação matemática

Rodrigo Cesar Arboleda - 10416722
Marcelo Isaias de Moraes Junior - 10550218
Igor Takeo Ambo de Melo - 10830054
Mateus Ferreira Gomes - 10734773
Guilherme Targon Marques Barcellos - 10724181

Modelagem

O problema consiste em encontrar um caminho com origem na primeira galáxia para percorrer todas as galáxias sendo que deve se percorrer o menor caminho e não deve passar pela mesma galáxia 2 vezes. Para resolver o problema iremos primeiramente modelar o problema.

Variáveis:

x_{ij} : variável inteira binária que indica se o caminho da galáxia ***i*** à galáxia ***j*** foi escolhido.

$x_{ij} \, \varepsilon \, \{0,1\}$ para todo $i \, \varepsilon \{1,2,..,n\}$ e $j \, \varepsilon \{1,2,..,n\}$ com $i \neq j$, sendo ***n*** o número de galáxias do problema em questão. Assume o valor 1 caso o caminho seja escolhido e 0 caso contrário.

d_{ij} : distância (custo) do caminho da galáxia ***i*** à ***j*** , em que $d_{ij} > 0$ para todo $i \, \varepsilon \{1,2,..,n\}$ e $j \, \varepsilon \{1,2,..,n\}$. Além disso, a distância de ***i*** à ***j*** é igual à distância de ***j*** à ***i***, isto é $d_{ij} = d_{ji}$.

Função objetivo

A função que queremos minimizar é a seguinte:

$$\min \sum_{i} \sum_{j} d_{ij} x_{ij}$$

Esta função irá retornar a distância percorrida para se passar por todas as galáxias.

Restrições

O problema apresenta algumas restrições. Primeiro precisamos garantir que iremos chegar em cada galáxia e também iremos sair dela, pois não podemos simplesmente “teletransportar” o telescópio para uma galáxia, além de que temos que passar por todas. Além disso iremos chegar e sair de cada galáxia apenas uma vez.

Para garantir que sempre haverá um caminho chegando a todas as galáxias e apenas uma vez, teremos a seguinte restrição:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, ..., n, i \neq j$$

Com isso conseguimos garantir que sempre iremos chegar em uma galáxia apenas uma vez.

Para garantir que iremos sair dessa galáxia e que também faremos isso uma vez apenas teremos a seguinte restrição:

$$\sum_{j=1}^{n} x_{ji} = 1 \qquad i = 1, ..., n, i \neq j$$

Dessa forma podemos garantir essas restrições. Por fim precisamos garantir outro ponto que é que não haverá subciclos, ou seja, dois ciclos independentes. Isso é um problema pois tal fato satisfaz as restrições anteriores mas não serve para o problema pois o telescópio precisaria “teletransportar” de um ciclo para o outro.

Para resolver este problema iremos criar mais uma restrição. A restrição que iremos criar é um pouco diferente do convencional, tal restrição se dá por contradição, ou seja, quando tivermos subciclo chegaremos em uma contradição e a restrição não irá valer invalidando a resposta. Tal método foi escolhido para facilitar a implementação do problema. Segue a restrição a seguir:

$$r_i - r_j + nx_{ij} \leq n - 1 \quad \forall i, j \in \{2, ..., n\}, i \neq j$$

$$r_i \geq 0 \quad i = 1, ..., n$$

Está restrição funciona da seguinte forma: n é o número de galáxias que temos, e com isso adicionamos uma variável r, esta variável irá servir apenas para gerar um inconsistência caso surja um ciclo. Para melhor entendimento de como a restrição funciona, segue um exemplo prático dela.

Tomamos 5 galáxias (1,2,3,4,5) onde a galáxia 1 está ligada com a 2, e a 2 com a 3, e a 3 com a 1. E por fim a galáxia 4 e 5 ligadas entre elas formando um subciclo. Ou seja, temos um ciclo em (1,2,3) e outro em (4,5).

Caso isso ocorra teremos que:

$$r_4 - r_5 + 5 * 1 \leq 4$$

$$r_5 - r_4 + 5 * 1 \leq 4$$

O que claramente é um absurdo. Esta formulação é conhecida como: Formulação de Miller-Tucker-Zemlin. A prova para esta formulação não será apresentada aqui pois pode ser facilmente encontrada.

Com isso nosso problema fica com a seguinte formulação:

Formulação

Função objetivo:

$$\min \sum_i \sum_j d_{ij} x_{ij}$$

Restrições:

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, ..., n \qquad (1)$$

$$\sum_{j=1}^{n} x_{ji} = 1 \quad i = 1, ..., n \qquad (2)$$

$$r_i - r_j + n x_{ij} \leq n - 1 \quad \forall i, j \in \{2, ..., n\}, i \neq j \qquad (3)$$

$$r_i \geq 0 \quad i = 1, ..., n \qquad (4)$$

Resolvendo um problema (toy problem)

Um problema simples para ser resolvido seria um problema com 5 galáxias (apesar de ter um número reduzido de galáxias, o problema exige muitos cálculos para gerar a resposta correta por conta da complexidade do problema), como modelamos o problema, caso conseguimos encontrar uma solução para o problema (mesma que não seja ótima) iremos conseguir utilizar o método simplex para encontrar a solução.

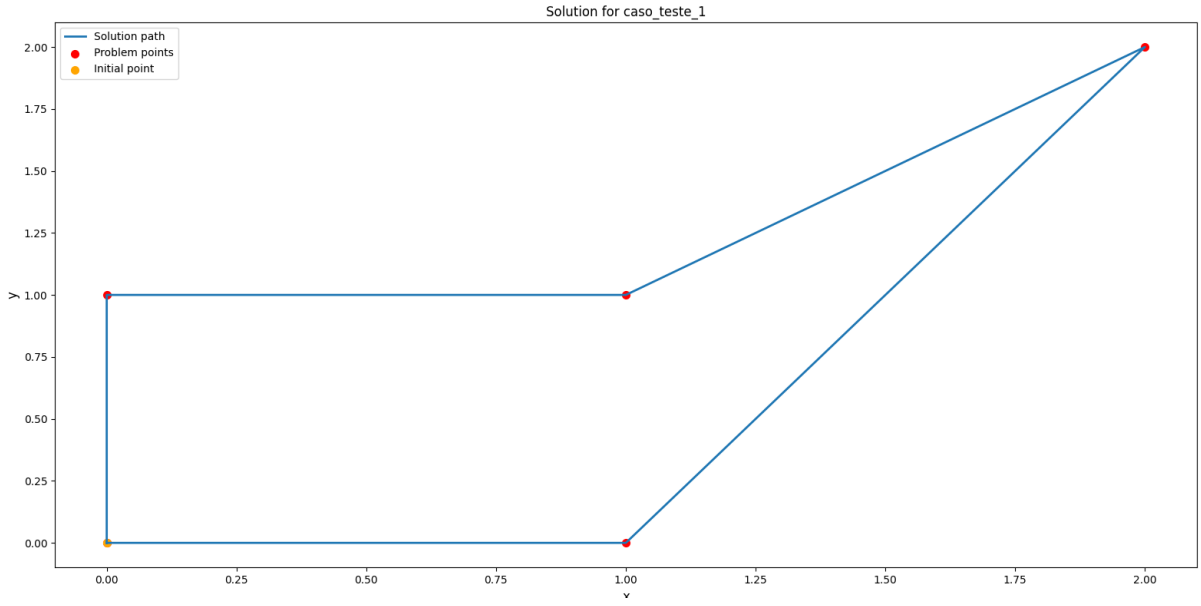
Um problema real seria com as seguintes galáxias:

- Galáxia 1: (0.0;0.0)
- Galáxia 2: (1.0;0.0)
- Galáxia 3: (0.0;1.0)
- Galáxia 4: (1.0;1.0)
- Galáxia 5: (2.0;2.0)

É fácil mostrar que conseguimos encontrar uma solução inicial para o problema. Uma forma seria ir de galáxia a galáxia na ordem dos pontos, ou seja, 1→ 2→ 3→ 4→ 5→ 1. Isso irá nos dar uma solução básica viável para o problema mas que não é ótima. Com isso basta aplicar o método simplex para resolver o problema.

Este problema ao ser executado no algoritmo que desenvolvemos que irá definir o modelo que criamos e buscar uma solução para o problema obtemos o seguinte resultado:

1 → 2 → 5 → 4 → 3 → 1, distância percorrida: 6.65.



Distance travelled: 6.650000

Podemos verificar de que fato esta solução é uma solução básica viável e ótima. Para isso basta verificar que satisfaz todas as restrições o que é trivial de verificar. Além disso, podemos aplicar o método simplex e verificar que não existe nenhuma direção para

caminhar que irá reduzir o valor da função objetivo, que como visto em aula, tal condição garante que encontramos solução ótima.

Outros exemplos

Também foram adicionados outros exemplos para serem executados. A lista com o nome dos arquivos com exemple segue a seguir:

- teste-1.tsp
- teste-2.tsp
- teste-3.tsp
- teste-4.tsp
- teste-5.tsp

Implementação

O algoritmo foi implementado em python 3 utilizando a biblioteca OR-Tools.

Leitura de entradas

As entradas foram definidas da mesma forma como os exemplos passados, por meio dos arquivos .tsp. Segue o exemplo do arquivo do *toy problem*.

```
NAME: caso_teste_1
COMMENT : Um caso teste simples feito pelo o grupo para o projeto
TYPE: TSP
DIMENSION: 5
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_SECTION
1 0.0 0.0
2 1.0 0.0
3 0.0 1.0
4 1.0 1.0
5 2.0 2.0
```

Ao executar o programa será pedido o nome do arquivo, basta digitar o nome do arquivo com a extensão que o programa irá carregar os dados do arquivo.
ATENÇÃO: O arquivo de entrada deve estar na mesma pasta do código a ser executado.

Matriz de distâncias

Ao receber as entradas, o programa cria uma matriz de distâncias na qual na linha ***i*** coluna ***j*** temos a distância entre a galáxia ***i*** e ***j.*** Após isso, essa matriz que inicialmente é calculada usando o tipo *float*, é convertida para uma matriz de inteiros da seguinte forma: Fazemos $M_{ij} * 10^n$ e o resultado convertemos para inteiro removendo as casas decimais. Isso faz com que a precisão seja limitada ao valor escolhido em ***n***.

Tal método foi escolhido por uma característica da biblioteca que não permite trabalhar com o tipo *float* na função objetivo. Tal alteração não limita o programa uma vez que o computador possui um limite de precisão mesmo para o tipo *float*, no nosso caso, a precisão é limitada pelo tamanho de um inteiro. Vale ressaltar que a resposta do problema não será alterada pois todas as distâncias são modificadas da mesma forma, só é preciso dividir a distância percorrida por 10^n para obter o valor real.

Definindo o problema

O problema é definido utilizando a biblioteca OR-TOOLS. Segue a definição de cada restrição do problema:

Restrição (1)
```
model.Add(sum([x[i][j] for j in range(n) if j != i]) == 1)
```

Restrição (2)
```
model.Add(sum([x[j][i] for j in range(n) if j != i]) == 1)
```

Restrição (3)
```
model.Add((y[i] - n * x[i][j]) >= y[j] - (n - 1))
```

Detalhes das variáveis como ***x*** só podendo assumir valor 0 ou 1 são definidos no momento de declarar a variável na biblioteca. Também foi limitado o tempo de execução do método para resolver o problema em 10 minutos para adiantar a parte 2. Outros detalhes de implementação podem ser vistos diretamente no código fornecido.

Mostrando respostas

A resposta do problema é mostrada utilizando a biblioteca Matplotlib, na qual desenha as galáxias em um plano cartesiano assim como o caminho a ser percorrido pelo telescópio. Além disso, também é mostrado no terminal a ordem dos pontos assim como a distância total percorrida.

Executando o código

Bibliotecas

- matplotlib
- numpy
- ortools
-

Para instalar as biblioteca basta utilizar o comando:
```
$ pip3 install <biblioteca>
```

Exemplo de execução (Linux)

```
$ python3 main.py
Insert test file name: teste-1.tsp
Solution path (Points visited):
[1, 2, 5, 4, 3, 1]
Distance travelled: 6.650000
```

Em caso de dúvida para executar o código, enviar um e-mail para algum membro do grupo.