# Projeto - Um Enigma das Galáxias

André Luís Mendes Fakhoury - 4482145 - andrefakhoury@usp.br
Débora Buzon da Silva - 10851687 - debora.buzon@usp.br
Gustavo Vinícius Vieira Silva Soares - 10734428 - gsoares@usp.br
Thiago Preischadt Pinheiro - 10723801 - thiagop@usp.br

SME0110 - Programação Matemática
Profas. Franklina Toledo e Marina Andretta

# 1 Modelagem

Queremos modelar o problema clássico do caixeiro viajante, em que cada vértice do grafo são galáxias. Deseja-se, portanto, encontrar alguma rota que visite todas as galáxias exatamente uma vez, e retorne à galáxia inicial, minimizando-se a distância total percorrida no trajeto. As galáxias são representadas por pontos no $\mathbb{R}^2$.

São constantes do problema:

$$\begin{aligned}
n &= \text{número de galáxias}\\
x_i &= \text{coordenada } x \text{ da galáxia } i\\
y_i &= \text{coordenada } y \text{ da galáxia } i\\
d_{ij} &= \text{distância euclidiana arredondada entre as galáxias } i \text{ e } j\\
i, j &\in \{1, \dots, n\}
\end{aligned}$$

São conhecidas as localizações das galáxias como pontos $(x, y)$ no plano cartesiano, e a distância euclidiana arredondada entre dois pontos é definida como:

$$d_{i,j} = round(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}) = \lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + 0.5 \rfloor$$

em que *round(x)* é o inteiro mais próximo de $x$, também podendo ser escrito como $\lfloor x + 0.5 \rfloor$, em que $\lfloor x \rfloor$ é a função piso.

## 1.1 Descrição das variáveis

$$p_{ij} = \begin{cases} 1, \text{ caso a aresta } (i, j) \text{ faz parte da rota} \\ 0, \text{ caso contrário} \end{cases}$$

## 1.2 Função objetivo

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} p_{ij}$$

Queremos minimizar a soma das distâncias, considerando apenas as arestas escolhidas para serem percorridas.

## 1.3 Restrições

$$
\begin{aligned}
p_{ii} &= 0 & i &= 1, \dots, n & (1)\\
\sum_{j=1}^{n} p_{ij} &= 1 & i &= 1, \dots, n & (2)\\
\sum_{j=1}^{n} p_{ji} &= 1 & i &= 1, \dots, n & (3)\\
\sum_{i,j \in S} p_{ij} &\leq |S| - 1 & S &\subseteq \{2, \dots, n\}, |S| \geq 2 & (4)\\
p_{ij} &\in \{0, 1\} & i, j &= 1, \dots, n & (5)
\end{aligned}
$$

Estas restrições garantem os seguintes itens:

1. Uma galáxia não pode ir para ela mesma;
2. Cada galáxia é visitada por outra apenas uma vez;
3. Cada galáxia visita outra apenas uma vez;
4. Há apenas um ciclo, que contém todos os nós. As restrições anteriores garantem que toda galáxia faz parte de um ciclo, se existe mais de um ciclo, pelo menos um deles é um subconjunto de $\{2, \dots, n\}$. Como um ciclo de tamanho $n$ contém exatamente $n$ arestas, limitar a quantidade de arestas entre esse subconjunto a $n - 1$ impede a formação do ciclo;
5. A variável $p_{ij}$ é binária.

# 2 Toy Problem

## 2.1 Exemplo

A figura 1 exemplifica um problema com 5 galáxias, descritas como $P1, P2, P3, P4, P5$.

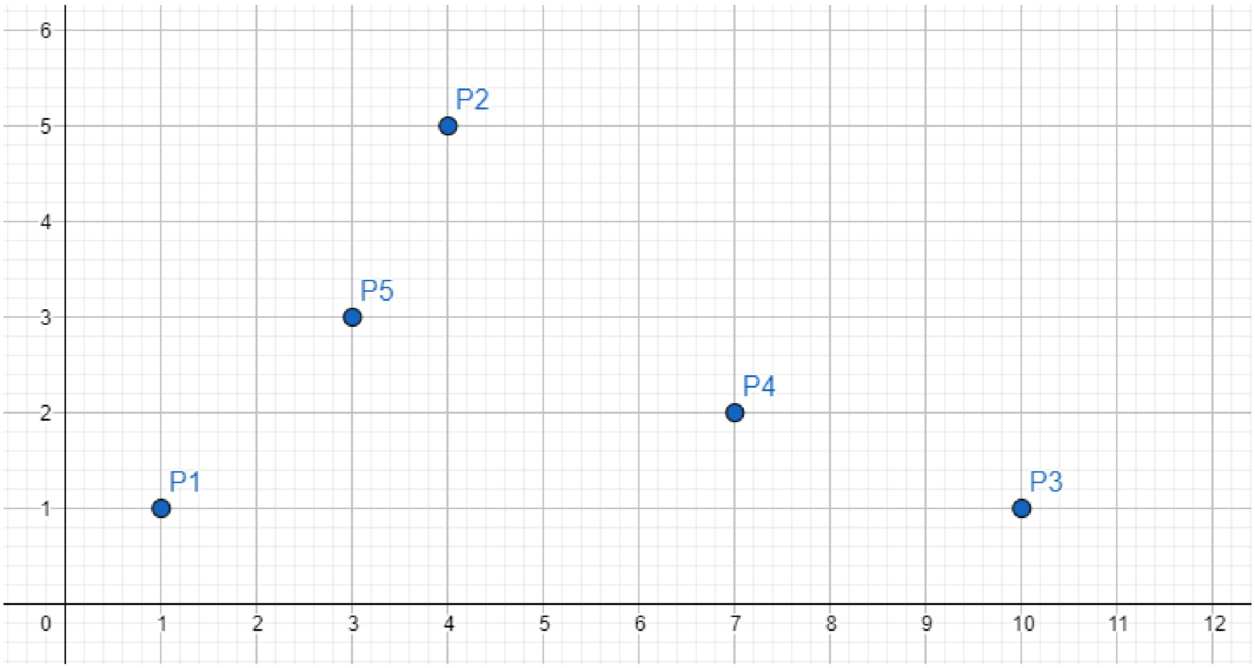



Figura 1: Exemplo de problema

Dele, conseguimos retirar os seguintes dados:

$$n = 5$$
$$x = [1.0, 3.0, 4.0, 7.0, 10.0]$$
$$y = [1.0, 3.0, 5.0, 2.0, 1.0]$$
$$d = \begin{bmatrix} 0.0 & 5.0 & 9.0 & 6.0 & 3.0 \\ 5.0 & 0.0 & 7.0 & 4.0 & 2.0 \\ 9.0 & 7.0 & 0.0 & 3.0 & 7.0 \\ 6.0 & 4.0 & 3.0 & 0.0 & 4.0 \\ 3.0 & 2.0 & 7.0 & 4.0 & 0.0 \end{bmatrix}$$

E estes dados são então utilizados na modelagem descrita anteriormente.

## 2.2 Solução

Uma possível solução pode ser encontrada na figura 2. Nela, a ordem encontrada de visita é $1 \to 5 \to 2 \to 4 \to 3 \to 1$, e o valor da função objetivo (distância total percorrida) é de 21.

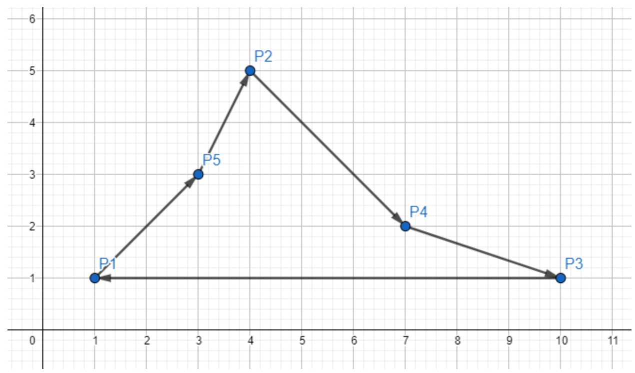



Figura 2: Solução do *Toy Problem*

Como no exemplo as arestas são bidirecionais, o programa para encontrar o melhor caminho também pode encontrar o caminho reverso (no caso, $1 \to 3 \to 4 \to 2 \to 5 \to 1$).

# 3 Mudanças e heurísticas

Visando a diminuição no tempo de execução e análise de eficiência da solução encontrada, foram pensadas algumas mudanças na solução *naive* original:

1. Ao invés de adicionar restrições impedindo todos os subciclos de serem formados (que seriam muitas restrições), foi feita a adição incremental destas restrições. Posteriormente, estes subciclos encontrados pelo *solver* foram conectados entre si de maneira gulosa: enquanto a quantidade de componentes (subciclos) existentes for maior que 1, é feita uma junção entre duas componentes, de forma que o custo das arestas adicionadas (junção das duas componentes) menos as arestas removidas (remoção de uma aresta de cada componente) seja minimizado.

2. Foi calculada uma solução viável inicial, de forma gulosa, e enviada ao *Solver*. Esta solução se baseia em: começar do vértice inicial 1, e ir para o vértice mais próximo a ele que ainda não foi visitado. Posteriormente, procura-se o vértice (não visitado) mais próximo a este último encontrado, e assim por diante. Estes passos se repetem até que todas as galáxias já tenham sido visitadas.

3. Com o objetivo de comparação e análise de heurística gulosa utilizada, foram calculadas também soluções viáveis aleatórias (permutações aleatórias) e enviadas ao *Solver*. Posteriormente será feita a comparação entre as duas heurísticas.

4. O tempo limite informado ao solver foi de 30 minutos por solução. Como o algoritmo desenvolvido funciona por iterações, cada iteração recebe uma parcela destes 30 minutos. A escolha da quantidade desta parcela leva em consideração a iteração que está

sendo executada (iterações iniciais necessitam de menos tempo para encontrarem uma solução, enquanto que iterações posteriores necessitam de mais tempo).

5. A fim de melhorar as soluções tanto utilizando a heurística gulosa, quanto a randômica, foi aplicado o Algoritmo *2-OPT* nas soluções finais, de forma a reduzir o número de arestas que cruzassem os próprios caminhos e reduzir o custo total do ciclo. Essa mudança gerou diferença significativa para a instância Uruguay, que continha o maior número de cidades.

# 4 Instâncias Analisadas

O problema foi analisado nas instâncias *Western Sahara*, *Djibouti*, *Qatar* e *Uruguay* da biblioteca Waterloo [1], além de no *Toy Problem* descrito inicialmente. A tabela 1 contém alguns dados de suas execuções, utilizando a heurística **gulosa** para solução viável inicial. Os atributos analisados são: nome da instância, solução encontrada pelo algoritmo, solução "ótima" (melhor solução conhecida) informada no site original [1], número da iteração em que foi encontrada a melhor solução, quantidade de nós visitados pelo *Branch-and-Cut*, tempo de execução e *GAP* relativo.

O GAP relativo foi calculado da seguinte forma:

$$\text{GAP} = \frac{\text{LPencontrado - LPsite}}{\text{LPsite}} \cdot 100\%$$

em que *LPencontrado* foi a solução encontrada (que é um limitante primal do problema) e *LPsite* é a solução descrita em [1] (que também é um limitante primal).

| **Instância** | **Encontrado** | **Ótimo** | **Iter.** | **Qt. Nós** | **Tempo (s)** | **GAP (%)** |
|---|---|---|---|---|---|---|
| Toy Problem | 21 | 21 | 1 | 1 | 0.01 | 0 |
| Western Sahara | 27603 | 27603 | 4 | 4 | 0.3 | 0 |
| Djibouti | 6656 | 6656 | 2 | 4 | 0.67 | 0 |
| Qatar | 9352 | 9352 | 13 | 73 | 458.0 | 0 |
| Uruguay | 90957 | 79114 | 1 | 1 | 120.21 | 14.96 |

Tabela 1: Dados das execuções das instâncias com heurística gulosa

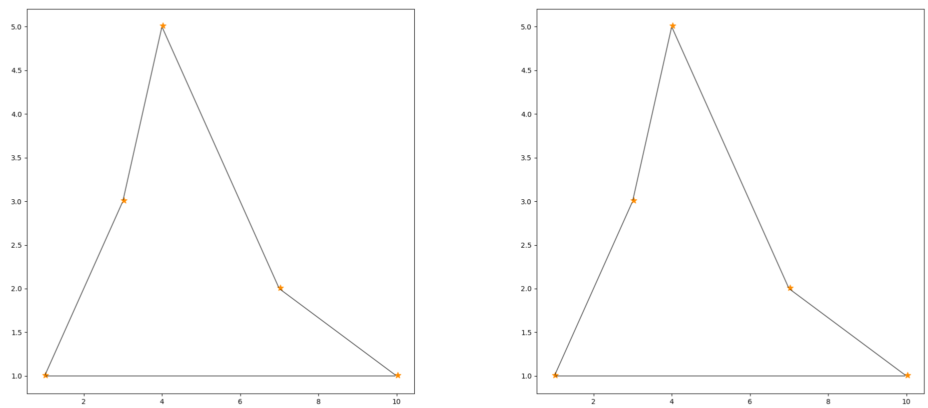
Utilizando a heurística **randômica** para solução viável inicial, os dados da tabela 2 foram obtidos:

| **Instância** | **Encontrado** | **Ótimo** | **Iter.** | **Qt. Nós** | **Tempo (s)** | **GAP (%)** |
|---|---|---|---|---|---|---|
| Toy Problem | 21 | 21 | 1 | 1 | 0.01 | 0 |
| Western Sahara | 27603 | 27603 | 4 | 4 | 0.29 | 0 |
| Djibouti | 6656 | 6656 | 4 | 4 | 0.56 | 0 |
| Qatar | 9352 | 9352 | 13 | 73 | 400.17 | 0 |
| Uruguay | 90444 | 79114 | 2 | 3 | 335.38 | 14.32 |

Tabela 2: Dados das execuções das instâncias com heurística randômica

Com isso, percebe-se que a heurística empregada no valor inicial enviado ao *solver* não ocasionou grandes mudanças, significativas, no resultado obtido. Outro ponto interessante é que as soluções encontradas nas instâncias *Western Sahara, Djibouti e Qatar* se equivalem às melhores citadas em [1].

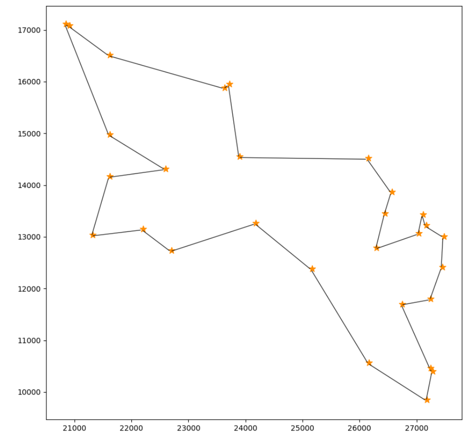
Os ciclos encontrados na resolução do *Toy problem* podem ser vistos na figura 3. Pode-se perceber, tanto pelas tabelas quanto pela figura que o *solver* não teve nenhum problema em encontrar a solução neste caso.



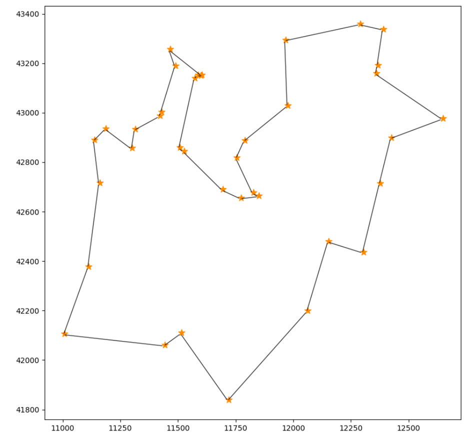

(a) Heurística gulosa (b) Heurística randômica

Figura 3: Ciclos encontrados na instância do *Toy Problem*

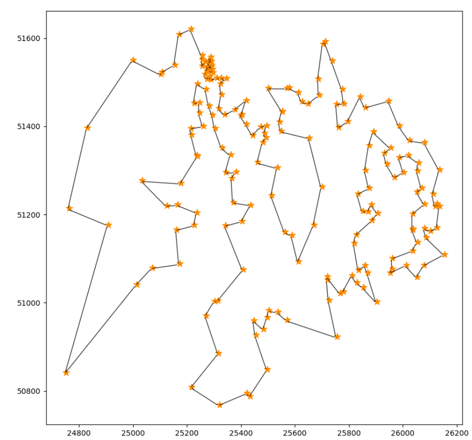
Um esquema gráfico dos ciclos encontrados pela heurística gulosa pode ser visualizado na figura 4.



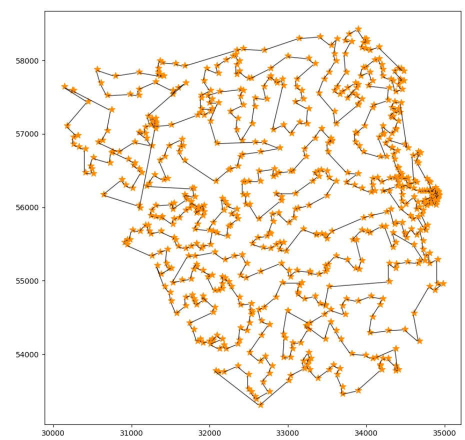

(a) Western Sahara




(b) Djibouti



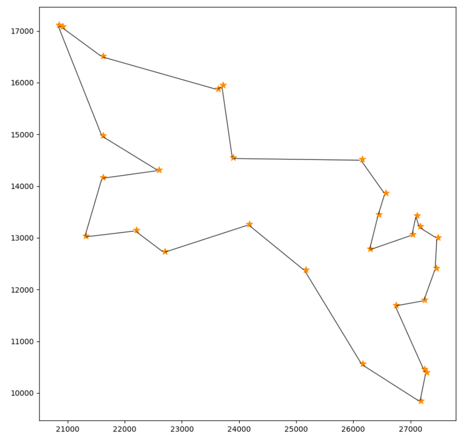

(c) Qatar



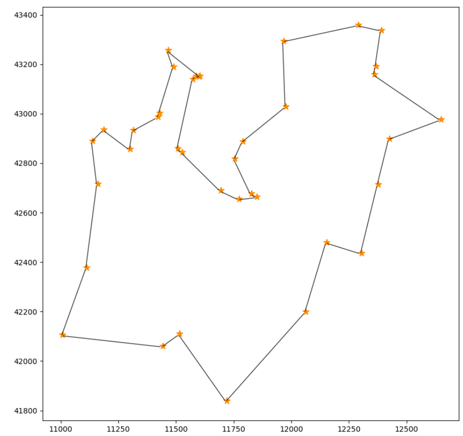

(d) Uruguay

Figura 4: Ciclos encontrados em cada instância, com heurística gulosa

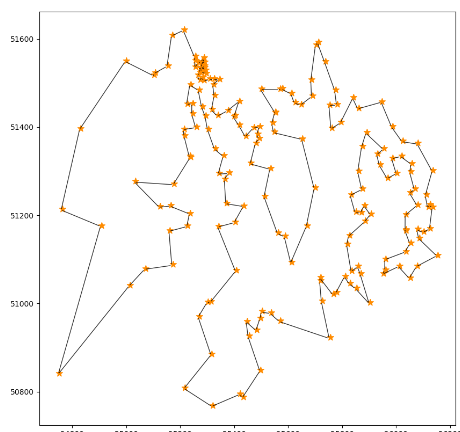
Já os ciclos encontrados com a utilização de heurística randômica podem ser visualizados na figura 5.



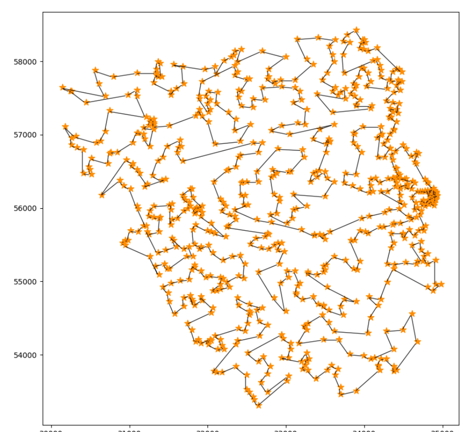

(a) Western Sahara




(b) Djibouti




(c) Qatar




(d) Uruguay

Figura 5: Ciclos encontrados em cada instância, com heurística randômica

Visualizando os ciclos, também pode-se analisar (agora de maneira visual) que as heurísticas de valor inicial não causaram grandes diferença na solução obtida.

Nas figuras 4 e 5, pode-se perceber, visualmente, que a solução encontrada na instância Uruguay não possui arestas que cruzem o próprio caminho, dado que foi utilizado a heurística *2-OPT* para removê-las. Sem a utilização do *2-OPT*, apenas com a heurística gulosa, a melhor solução encontrada possuiu custo 99881 e pode ser vista na figura 6.

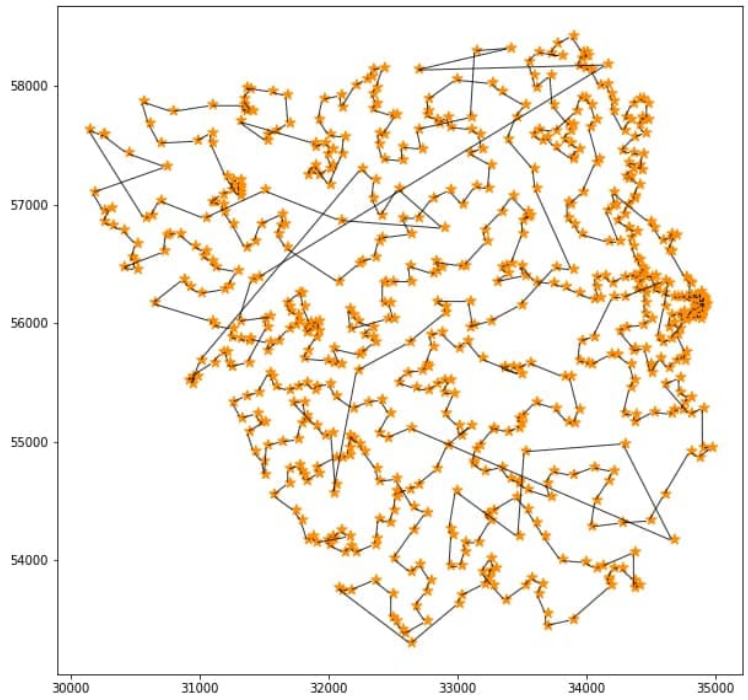



Figura 6: Solução da instância Uruguay sem o uso da heurística *2-OPT*

# 5 Outras Instâncias

Também foram executadas outras instâncias, do mesmo jeito que as anteriores. Da mesma referência [1], foram executadas as instâncias *Luxemburgo, Rwanda e Zimbabwe*.

As informações de execução destas instâncias com as heurísticas gulosa e *2-Opt* podem ser vistas na tabela 3.

| **Instância** | **Encontrado** | **Ótimo** | **Iter.** | **Qt. Nós** | **Tempo (s)** | **GAP (%)** |
|---|---|---|---|---|---|---|
| Luxemburgo | 12589 | 11340 | 1 | 1 | 95.88 | 11 |
| Rwanda | 30022 | 26051 | 2 | 2 | 268.24 | 15.21 |
| Zimbabwe | 107885 | 95345 | 1 | 1 | 169.6 | 13.15 |

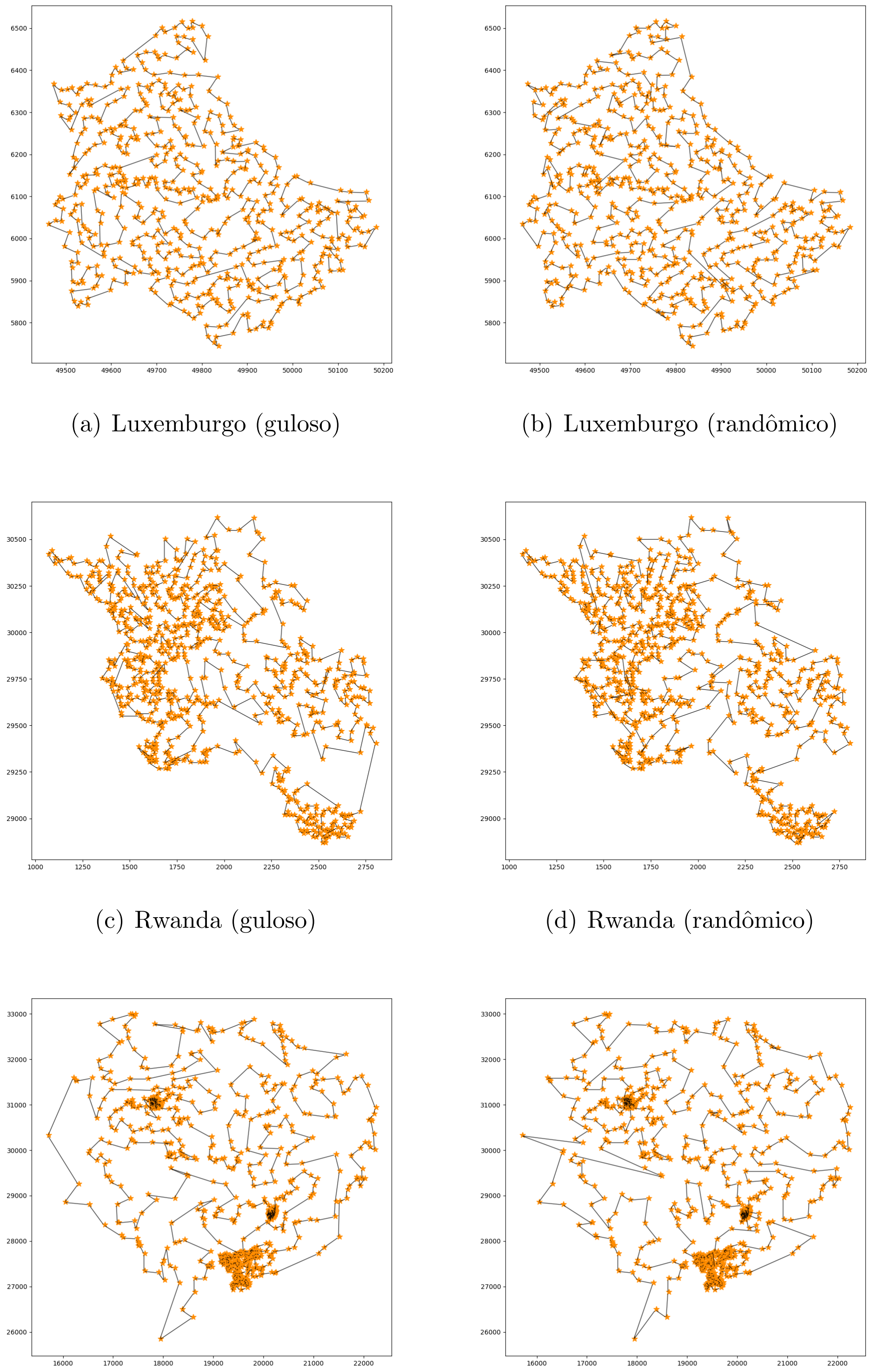
Tabela 3: Dados das execuções das instâncias com heurística gulosa

As informações de execução destas instâncias com as heurísticas randômica e *2-Opt* podem ser vistas na tabela 4.

| **Instância** | **Encontrado** | **Ótimo** | **Iter.** | **Qt. Nós** | **Tempo (s)** | **GAP (%)** |
|---|---|---|---|---|---|---|
| Luxemburgo | 13027 | 11340 | 1 | 3 | 301.42 | 14.87 |
| Rwanda | 30618 | 26051 | 4 | 6 | 854.47 | 17.53 |
| Zimbabwe | 107822 | 95345 | 1 | 7 | 1321.01 | 13.08 |

Tabela 4: Dados das execuções das instâncias com heurística randômica

Nestas instâncias, assim como anteriormente, a diferença entre as heurísticas não foi tão extraordinária. Na figura 7 podem ser vistos os ciclos encontrados.



(a) Luxemburgo (guloso)

(b) Luxemburgo (randômico)

(c) Rwanda (guloso)

(d) Rwanda (randômico)

(e) Zimbabwe (guloso)

(f) Zimbabwe (randômico)

Figura 7: Ciclos encontrados em cada instância

# 6 Implementação

O projeto foi implementado em *Python*, utilizando-se a biblioteca *OR-Tools* [2]. No arquivo fonte (extensão *.py*) está a documentação do código desenvolvido.

# Referências


[1] U. Waterloo, "National traveling salesman problems," 2020. Disponível em: `http://www.math.uwaterloo.ca/tsp/world/countries.html`.

[2] Google, "Or-tools." Disponível em: `https://developers.google.com/optimization/reference/python/`.

[3] F. Toledo, *Aula 6 Modelagem Var. Inteiras*. 2020.

[4] S. Carlson, "Algorithm of the gods," 1997.

[5] Google, "Google colab." Disponível em: `https://colab.research.google.com/`.

[6] Deepnote, "Deep note." Disponível em: `https://deepnote.com/`.