UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E COMPUTAÇÃO

LUCAS G. M. MIRANDA - 10265892 - lucasgmm@usp.br

MARCELA TIEMI SHINZATO - 10276953 - marcelats@usp.br

SÉRGIO RICARDO G. B. FILHO - 10408386 - sergiobarbosa@usp.br

TIAGO LASCALA AUDE - 8936742 - tiago.aude@usp.br

SME0110 – PROGRAMAÇÃO MATEMÁTICA

PROFESSORA MARINA ANDRETTA

PROFESSORA FRANKLINA TOLEDO

RELATÓRIO 1

São Carlos – SP

2020

QUESTÃO 1: escrever um modelo em linguagem de modelagem

Função objetivo:

Similar ao problema do caixeiro viajante, desejamos minimizar o ângulo total o qual o telescópio se desloca para observar todas as galáxias uma vez cada e retornar à galáxia inicial. Podemos simplificar a determinação do ângulo ao considerarmos as distâncias euclidianas entre as galáxias.

$$min \sum_{i,j \in A} \left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} * Y_{ij}\right)$$

Variáveis:

Para isso, usamos uma variável binária Y_{ij} para cada par de galáxia cujos índices pertencem ao conjunto A, que assume valor 1 se o telescópio sai da galáxia *i* e vai para a galáxia *j* e assume valor 0 caso contrário.

$Y_{ij} = \{0, 1\}$

$i,j \in A$

Parâmetros constantes:

Usaremos um referencial de coordenadas retangulares sendo o ponto fixo do telescópio a origem do sistema e as galáxias sendo dadas por pontos fixos, portanto d não é variável pois possui valor constante para cada par de galáxia.

Para extrair as informações das distâncias euclidianas pelos mapas fornecidos na biblioteca Waterloo, vamos determinar *d* medindo as distâncias no mapa entre as coordenadas x,y das galáxias em relação à origem.

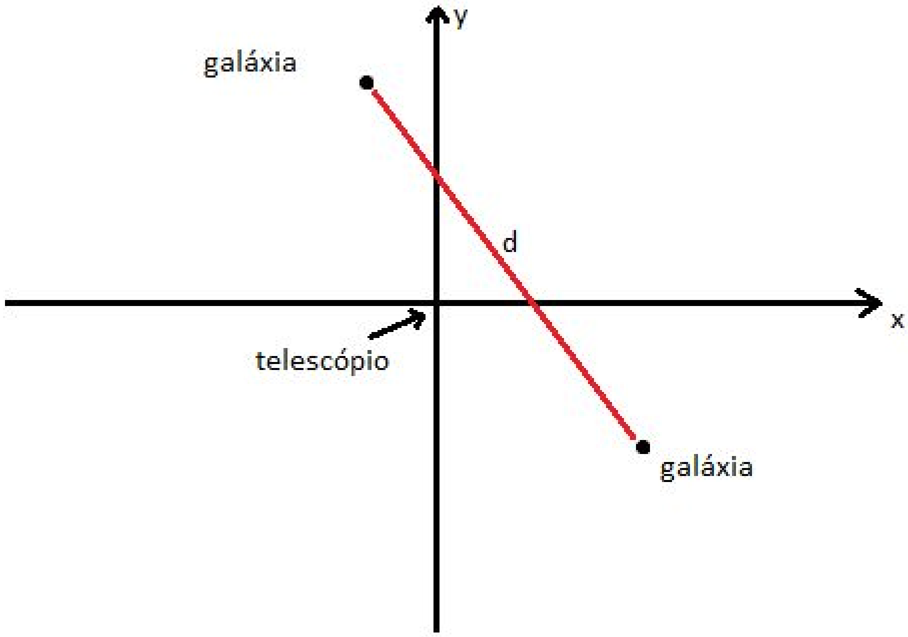

Figura 1: exemplo de representação das galáxias usando coordenadas retangulares e distância euclidiana

Restrições:

Além das restrições de domínios das variáveis, teremos as mesmas restrições do problema do caixeiro viajante: visitar uma vez cada galáxia e retornar à inicial, o que implica proibir subciclos ilegais.

A seguinte restrição garante que, para cada galáxia **i**, apenas uma galáxia a sucederá no caminho tomado:

$\sum_{j=1}^{n} Y_{ij} = 1, i = 1, ..., n$, n: número de galáxias a serem observadas

A seguinte restrição garante que para cada galáxia **i** apenas uma galáxia a antecede:

$\sum_{j=1}^{n} Y_{ji} = 1, i = 1, ..., n$, n: número de galáxias a serem observadas

Para proibir os subciclos ilegais, consideramos que **N** é o conjunto de galáxias, portanto cada subciclo **S** precisa estar contido em **N**, excluindo a primeira galáxia, pois o ciclo final (da solução ótima) contém a primeira galáxia, portanto, não podemos adicionar uma restrição que o proíba:

$S \subseteq N - \{1\}$

Definimos que um subciclo tem no mínimo 2 galáxias (embora isso seja relaxado na implementação em *Python* do modelo, para facilitar a detecção de autoloops):

$|S| \geq 2$

Por fim, definimos que em todo subciclo o número de arestas é menor que o número de vértices. Isso garante que o subciclo não se feche, pois o único ciclo que queremos que seja fechado é o que contém todas as galáxias.

$$\sum_{i,j \in S} y_{ij} \leq |S| - 1$$

Modelo final:

$$min \sum_{i,j \in A} (\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} * Y_{ij})$$

S.A.:

$Y_{ij} = \{0, 1\}$

$i,j \in A$

$$\sum_{j=1}^{n} Y_{ij} = 1, i = 1, ..., n$$

$$\sum_{j=1}^{n} Y_{ji} = 1, i = 1, ..., n$$

$S \subseteq N - \{1\}$

$|S| \geq 2$

$$\sum_{i,j \in S} y_{ij} \leq |S| - 1$$

QUESTÃO 2: resolução do *toy problem*

Adicionando restrições ao problema

Será resolvido um *toy problem* com cinco galáxias. Sua representação pictórica está posicionadas abaixo. Note que, nela, cada galáxia é representada por um símbolo de estrela, juntamente com suas coordenadas:

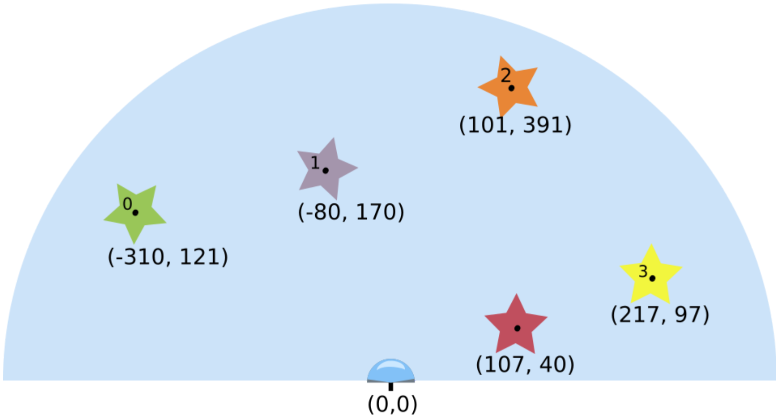

Figura 2: *toy problem* criado contendo 5 galáxias

A solução do problema será feita através do código em anexo. Esse código, em essência, executa a seguinte sequência de passos:

1. Implementa o modelo descrito sem as restrições de eliminação de subciclos ilegais;
2. Resolve-o;
3. Se a solução encontrada não possuir subciclos ilegais, então ela é ótima, e corresponde à resposta final;
4. Se a solução possuir subciclos ilegais, adiciona ao modelo as restrições de eliminação desses subciclos;
5. Resolve o problema novamente usando o modelo com as novas restrições e volta ao passo 2.

A detecção de subciclos ilegais no passo 2 é feita através da execução do algoritmo de busca em largura, implementado pela função `acha_subciclos`.

As restrições de eliminação de subciclos ilegais foram adicionadas dessa maneira para evitar gerar todas elas de uma vez só, visto que isso é custoso para casos em que o número de vértices é muito elevado. Mas, de qualquer forma, verificou-se que isso não ajudou tanto como se pensava, pois ainda é necessário resolver um problema de programação inteira a cada iteração (passos 2-5). Apesar disso para um número pequeno de vértices, como é o caso do *toy problem,* o algoritmo funciona de maneira rápida.

Resolvendo o toy problem

A seguir, vamos detalhar a resolução do toy problem. A primeira iteração resolve o problema sem nenhuma restrição de eliminação de subciclos ilegais:

$$min \sum_{i,j \in A} (\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} * Y_{ij})$$

S.A.:

$$Y_{ij} = \{0, 1\}$$

$$i,j \in A$$

$$\sum_{j=1}^{n} Y_{ij} = 1, i = 1, ..., n$$

$$\sum_{j=1}^{n} Y_{ji} = 1, i = 1, ..., n$$

A seguinte solução é retornada:

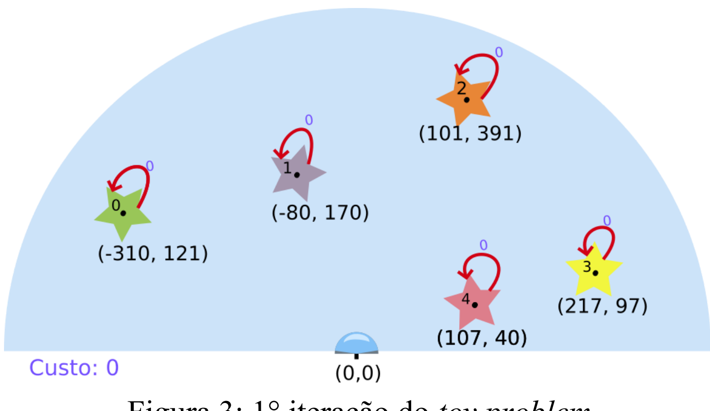

Figura 3: 1° iteração do *toy problem*

Ela é inválida, porque temos auto-loops (arestas que conectam um vértice a ele mesmo). O algoritmo detecta auto-loops através da mesma função que detecta subciclos ilegais (`acha_subciclos`), porque ela retorna as componentes conexas que não incluem o vértice 0 de um dado grafo. Dessa forma, os auto-loops detectados pelo algoritmo são {1}, {2}, {3} e {4}.

Para proibir que soluções com esses mesmos auto-loops sejam retornadas na próxima iteração, adicionamos as seguintes restrições:

- $Y_{ii} \leq 0$, i = 1,...,4

Resolvendo o problema novamente com essas restrições, obtemos a seguinte solução:

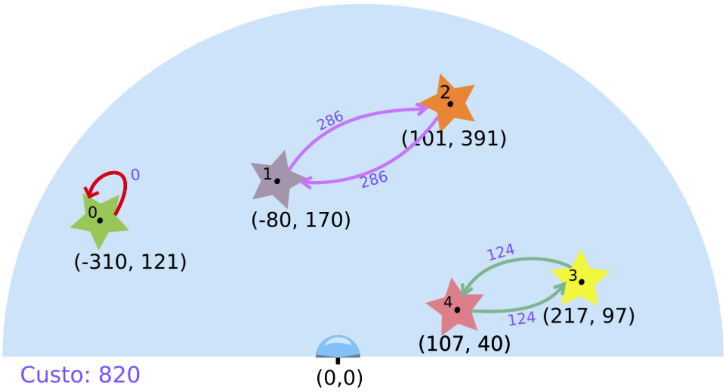

Figura 4: 2° iteração do *toy problem*

Ela é inválida porque existe um autoloop e dois subciclos ilegais. No entanto, o algoritmo detecta apenas os subciclos {1, 2} e {4, 3}. Para que a próxima solução não contenha esses subciclos, as seguintes restrições são adicionadas:

- $\sum_{i,j \in \{1,2\}} y_{ij} \leq |\{1,\ 2\}| - 1 = 2 - 1 = 1$
- $\sum_{i,j \in \{4,3\}} y_{ij} \leq |\{4,\ 3\}| - 1 = 2 - 1 = 1$

Resolvendo o problema novamente com essas restrições, obtemos a seguinte solução:

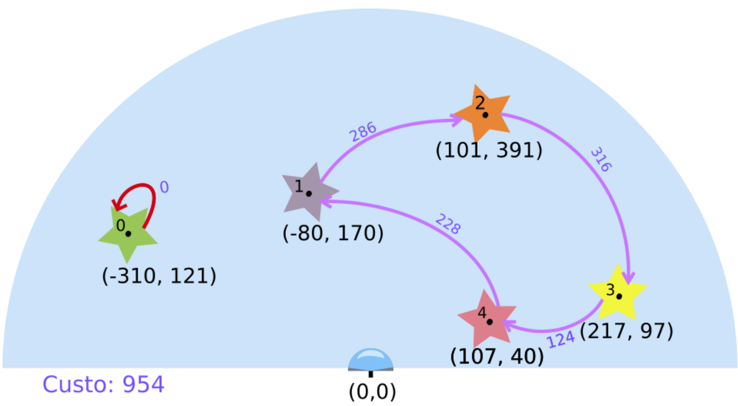

Figura 5: 3° iteração do *toy problem*

Ela é inválida porque existe um autoloop e um subciclo ilegal. No entanto, o algoritmo detecta apenas o subciclo $\{1,\ 2,\ 3,\ 4\}$. Para que a próxima solução não contenha esses subciclos, as seguinte restrição é adicionada:

- $\sum_{i,j \in \{1,2,3,4\}} y_{ij} \leq |\{1,\ 2,\ 3,\ 4\}| - 1 = 4 - 1 = 3$

Resolvendo o problema novamente com essa restrição, obtemos a seguinte solução:

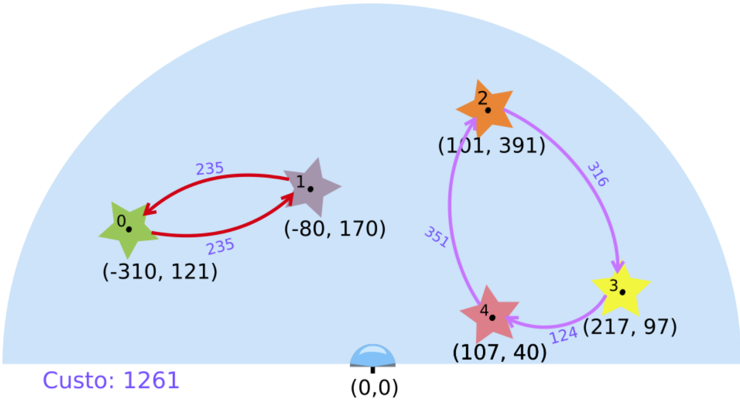

Figura 6: 4° iteração do *toy problem*

Ela é inválida porque existem dois subciclos ilegais. algoritmo detecta apenas o subciclo $\{2,\ 3,\ 4\}$, pois ele não contém a aresta 0. Para que a próxima solução não contenha esse subciclo, as seguinte restrição é adicionada:

- $\sum_{i,j \in \{2,3,4\}} y_{ij} \leq |\{2,\ 3,\ 4\}| - 1 = 3 - 1 = 2$

Resolvendo o problema novamente com essa restrição, obtemos a seguinte solução:

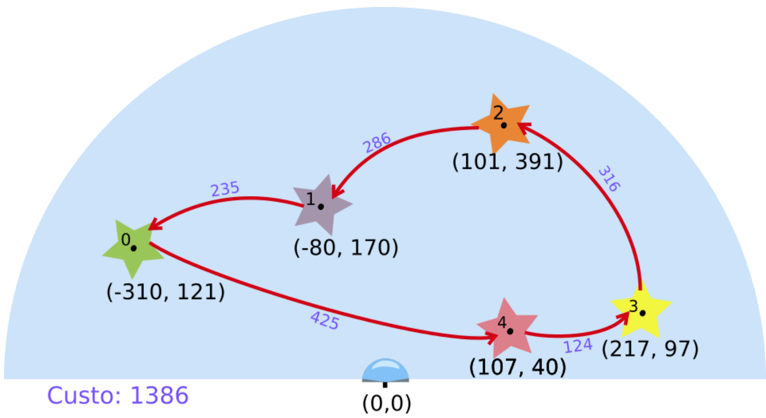

Figura 7: 5° iteração do *toy problem*

Como essa solução não possui nenhum autoloop, ou subciclo ilegal, ela é a solução ótima para o problema.

Referências bibliográficas

TOLEDO, FRANKLINA M. B. Programação Matemática, 2020. 46 slides. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/5645585/mod_resource/content/9/Aula06_Modelagem_Inteiros.pdf>. Acesso em: 10/10/2020

DE GIOVANNI, L.; DI SUMMA, M. Methods and Models for Combinatorial Optimization - Exact methods for the Traveling Salesman Problem. Disponível em: <https://www.math.unipd.it/~luigi/courses/metmodoc1819/m08.01.TSPexact.en.pdf>. Acesso em: 10/10/2020