Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

INVESTIGAÇÃO OPERACIONAL

2021/2022

DRONE – Caminho *Euleriano*

GRUPO:

A93290 - Joana Alves
A93264 - Maria Cunha
A93296 - Vicente Moreira

Data de Entrega: 24/03/2022

Índice

Índice de Figuras

Introdução

Este projeto tem como objetivo aprofundar os nossos conhecimentos obtidos nas aulas teóricas através da realização prática de um exercício de minimização. Assim, foi-nos apresentado um problema onde um veículo não tripulado (*drone*) tem de inspecionar linhas de transporte de energia elétrica para verificar se há vegetação a interferir com as mesmas. Este *drone* irá percorrer um mapa de arestas pré-definido, sendo o objetivo percorrer todas estas, minimizando a distância total percorrida.

Para além disso, é pedido que o *drone* comece a sua inspeção das linhas num determinado ponto da linha (Ponto I), tendo de terminar o seu percurso num ponto final (Ponto F). O *drone* também é capaz de se reposicionar para recomeçar a inspeção noutra aresta através do ar, não tendo, por isso, de seguir as linhas de alta tensão, sendo a distância entre os dois pontos, neste caso, calculada através da distância euclidiana.

Segundo o enunciado do trabalho prático, para obtermos o mapa final de linhas de alta tensão com as várias deslocações possíveis do *drone*, necessitamos de retirar algumas das arestas do conjunto de arestas BCDE de acordo com algumas restrições. Sendo 93296 o maior número de estudante do grupo, obtivemos como resultado a remoção das arestas C e E. Como consequência chegamos ao mapa seguinte (objeto de estudo deste relatório):

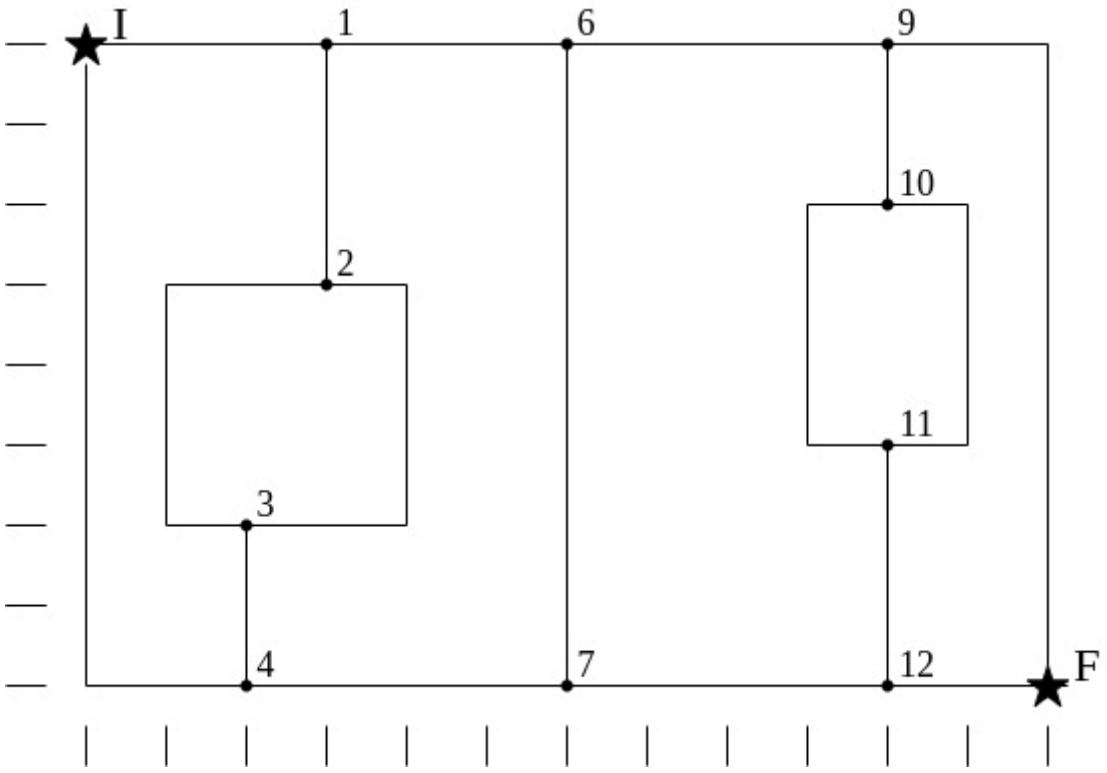

Figura 1 – Mapa Final de Linhas de Alta Tensão

Formulação do Problema

O problema descrito trata-se da resolução de um caminho *Euleriano*, onde é necessário obter um mapa onde os vértices de partida e chegada contenham um número ímpar de arestas e os restantes vértices um número par de arestas, sendo assim possível percorrer todo o mapa apenas uma vez, começando no ponto Inicial e terminando no ponto Final. Dividimos assim o problema em duas fases:

A primeira fase será de otimização do caminho a ser percorrido. Visto que não é permitido retirar arestas do mapa, teremos que "duplicá-las" ou recorrer a novas linhas, de forma a obter as condições necessárias para o caminho *Euleriano*, tendo de minimizar a distância percorrida por estes caminhos extras.

A segunda fase será a escolha do percurso a ser efetuado. Após o mapa "final" com o caminho Euleriano ser determinado, é necessário escolher um percurso que percorra todo este mapa.

Dado que o objetivo do problema será a "escolha" dos caminhos extras a serem efetuados, decidimos que as variáveis de decisão representariam o número de vezes que uma dada aresta ou linha (distância euclidiana entre os vértices) foi percorrida (para além do caminho normal). Logo, criamos uma variável por cada linha/aresta única no mapa, tendo esta de ser um número inteiro positivo. (Nota: Para evitar um número extenso de variáveis, consideramos que, por exemplo, a variável "linha1_2" é igual à "linha2_1", sendo apenas uma delas declarada.) (Nota: Em alguns casos, a distância de uma aresta entre dois vértices irá corresponder à sua distância euclidiana (linha). Neste caso ambos são declarados pois, apesar de serem redundantes, facilitam o processo de declaração de variáveis.)

Relativamente às restrições do problema, estas foram desenvolvidas de acordo com a definição de caminho Euleriano referido anteriormente, ou seja, os vértices Inicial (I) e Final (F) têm de ter um número ímpar de arestas/linhas, enquanto todos os outros vértices (vértices interiores) terão de ter um número par.

Assim, o problema tem como objetivo determinar o percurso em que todas as linhas de alta tensão são percorridas pelo menos uma vez, minimizando a distância total percorrida. Este objetivo será alcançado através da minimização da distância extra percorrida, visto que as linhas do mapa de inspeção obrigatória irão ter sempre um valor de distância fixo.

Por fim, conseguimos obter uma função objetivo linear uma vez que as variáveis de decisão presentes na mesma têm grau um pois apenas representam o número de vezes que uma dada aresta ou linha é percorrida ao longo do caminho. Para além disto, no que toca às restrições, estas também são lineares, uma vez que apenas definem a paridade de cada vértice, ou seja, igualam cada vértice a uma condição de paridade, utilizando números inteiros.

Modelo do Problema

Variáveis de Decisão

Seguindo a linha de pensamento apresentada acima, apresentamos então as definições das variáveis de decisão do nosso problema assim como os seus tipos:

$\boldsymbol{linhaX\_Y}$: $nº\ de\ vezes\ que\ a\ ligação\ direta\ do\ vértice\ X\ ao\ vértice\ Y\ pelo\ ar$ é $percorrida$

$\boldsymbol{arestaX\_Y}$: $nº\ de\ vezes\ que\ o\ caminho\ do\ vértice\ X\ ao\ vértice\ Y\ pelo\ mapa$ é $percorrido$

$$\boldsymbol{linhaX\_Y}, \boldsymbol{arestaX\_Y} \in IN_0 \quad e \quad \boldsymbol{X}, \boldsymbol{Y} \in \{I, F, 1,2,3,4,6,7,9,10,11,12\}$$

$$linhaX\_Y = linhaY\_X;\ arestaX\_Y = arestaY\_X$$

Parâmetros

De acordo com os dados presentes no enunciado do problema, conseguimos distinguir os seguintes parâmetros, ou seja, dados inalteráveis no decorrer da execução do problema:

- Distância euclidiana entre os vários vértices.
- Distância entre os vértices pelas linhas de alta tensão.

Função Objetivo

Como referido na formulação do problema, a função objetivo trata-se de um problema de minimização de distância extra percorrida. Assim, obtivemos a seguinte expressão onde as variáveis *custoExtraArestas* e *custoExtraLinhas* representam o somatório das distâncias extras percorridas quer pelas arestas do mapa quer pela distância euclidiana entre vértices, respetivamente. Para facilitar a compreensão das variáveis, uma vez que são ambas extensas, apresentamos a definição de cada uma destas de forma reduzida, estando o seu conteúdo total presente na secção seguinte através dos *prints* do ficheiro de *input* (Ficheiros).

$$\textbf{min}: custoExtraArestas + custoExtraLinhas$$

$$\boldsymbol{custoExtraArestas} = 3 \times arestaI\_1 + 10 \times arestaI\_4 + \cdots + 2 \times aresta12\_F$$

$$\boldsymbol{custoExtraLinhas} = 3 \times linhaI\_1 + 6 \times linhaI\_6 + \cdots + 2 \times linha12\_F$$

Restrições

No que toca às restrições, tal como mencionado anteriormente, estas definem a paridade do número de arestas/linhas de cada vértice, tendo os vértices Inicial e Final um número ímpar de arestas/linhas e os restantes vértices um número par.

Assim, definimos várias variáveis auxiliares para representar todas as arestas e linhas de um dado vértice, sendo a sua designação: $\boldsymbol{nodoX}$, $X \in \{I, F, 1,2,3,4,6,7,9,10,11,12\}$ e o seu conteúdo é a soma de todas as variáveis de decisão (linhas e arestas) que integram o vértice X.

Visto que, no contexto do nosso problema, todos os nodos requerem no mínimo apenas uma nova aresta/linha associada ao nodo, (nodo Inicial e Final têm número de arestas par quando devem ter um número ímpar, já os restantes têm número de arestas ímpares quando devem ter um número par) podemos restringir todos os nossos nodos para que apenas seja acrescentado um número ímpar de arestas, tendo um valor mínimo de 1, alterando assim a sua paridade.

Para isso, acrescentamos às restrições dos nodos que estes devem conter pelo menos uma nova aresta utilizando a equação $2k + 1,\ k \in \mathbb{N}_0$. Como cada nodo poderá ter quantidades de arestas/linhas extras diferentes, foi necessário criar uma instância de k para cada nodo presente no problema.

Apresentamos, então, as definições das várias restrições de forma resumida, estando, novamente, o seu conteúdo total presente na secção seguinte através dos *prints* do ficheiro de *input* (Ficheiros).

$$\boldsymbol{nodoI} = arestaI\_1 + arestaI\_4 + linhaI\_1 + linhaI\_6 + linhaI\_9 + linhaI\_2 + linhaI\_10 + linhaI\_3 + linhaI\_11 + linhaI\_4 + linhaI\_7 + linhaI\_12 + linhaI\_F;$$

…

$$\boldsymbol{nodoF} = aresta12\_F + aresta9\_F + linha12\_F + linha7\_F + linha4\_F + linha11\_F + linha3\_F + linha10\_F + linha2\_F + linha9\_F + linha6\_F + linha1\_F + linhaI\_F$$

$$\boldsymbol{nodoI} = 2 \times kI + 1$$

$$\boldsymbol{nodo1} = 2 \times k1 + 1$$

…

$$\boldsymbol{nodo12} = 2 \times k12 + 1$$

$$\boldsymbol{nodoF} = 2 \times kF + 1$$

$$\boldsymbol{kX} \in \mathbb{N}_0,\ X \in \{I, F, 1,2,3,4,6,7,9,10,11,12\}$$

Ficheiros

Ficheiro Input

```
/*
I __ __ __ 1 __ __ __ 6 __ __ __ __ 9 __ __
|          |          |             |       |
|          |          |          __ | __    |
|   __ __ | __       |          |  10   |  |
|   |      2    |     |          |       |  |        // Mapa Original Drone
|   |           |     |          |__ 11__|  |
|   |__ 3 __ __|      |             |       |
|       |            |             |       |
| __ __ | __ __ __ __ | __ __ __ __ | __ __|
        4            7            12       F
*/

// FUNÇÃO OBJETIVO
// Minimização da soma dos caminhos extra escolhidos.
min: custoExtraArestas + custoExtraLinhas;

// VARIÁVEIS DE DECISÃO
// Somatório do caminho extra percorrido pelas arestas.
custoExtraArestas = 3 arestaI_1 + 10 arestaI_4 + 3 aresta1_6 + 3 aresta1_2 + 4 aresta6_9 + 8 aresta6_7 + 2 aresta9_10 + 10 aresta9_F + 6 aresta2_3esq + 6 aresta2_3dir + 5 aresta10_11esq + 5 aresta10_11dir + 2 aresta3_4 + 3 aresta11_12 + 4 aresta4_7 + 4 aresta7_12 + 2 aresta12_F;

// Somatório do caminho extra percorrido pelas linhas.
custoExtraLinhas = 3 linhaI_1 + 6 linhaI_6 + 10 linhaI_9 + 4.24 linhaI_2 + 10.2 linhaI_10 + 6.32 linhaI_3 + 11.18 linhaI_11 + 8.25 linhaI_4 + 10 linhaI_7 + 12.81 linhaI_12 + 14.42 linhaI_F + 3 linha1_6 + 7 linha1_9 + 3 linha1_2 + 7.28 linha1_10 + 6.08 linha1_3 + 8.6 linha1_11 + 8.06 linha1_4 + 8.54 linha1_7 + 10.63 linha1_12 + 12.04 linha1_F + 4 linha6_9 + 4.24 linha6_2 + 4.47 linha6_10 + 7.21 linha6_3 + 6.4 linha6_11 + 8.94 linha6_4 + 8 linha6_7 + 8.94 linha6_12 + 10 linha6_F + 7.62 linha9_2 + 2 linha9_10 + 10 linha9_3 + 5 linha9_11 + 11.31 linha9_4 + 8.94 linha9_7 + 8 linha9_12 + 8.25 linha9_F + 7.07 linha2_10 + 3.16 linha2_3 + 7.28 linha2_11 + 5.1 linha2_4 + 5.83 linha2_7 + 8.6 linha2_12 + 10.3 linha2_F + 8.94 linha10_3 + 3 linha10_11 + 10 linha10_4 + 7.21 linha10_7 + 6 linha10_12 + 6.32 linha10_F + 8.06 linha3_11 + 2 linha3_4 + 4.47 linha3_7 + 8.25 linha3_12 + 10.2 linha3_F + 8.54 linha11_4 + 5 linha11_7 + 3 linha11_12 + 3.61 linha11_F + 4 linha4_7 + 8 linha4_12 + 10 linha4_F + 4 linha7_12 + 6 linha7_F + 2 linha12_F;

//Definição dos NODOS - Junção de todos os caminhos que percorrem um nodo.
//Exemplo - NodoI - Todos as arestas e linhas que passam no nodo I.
nodoI = arestaI_1 + arestaI_4 + linhaI_1 + linhaI_6 + linhaI_9 + linhaI_2 + linhaI_10 + linhaI_3 + linhaI_11 + linhaI_4 + linhaI_7 + linhaI_12 + linhaI_F;
nodo1 = aresta1_6 + aresta1_2 + linha1_6 + linha1_9 + linha1_2 + linha1_10 + linha1_3 + linha1_11 + linha1_4 + linha1_7 + linha1_12 + linha1_F + arestaI_1 + linhaI_1;
nodo6 = aresta6_9 + aresta6_7 + linha6_9 + linha6_2 + linha6_10 + linha6_3 + linha6_11 + linha6_4 + linha6_7 + linha6_12 + linha6_F + aresta1_6 + linha1_6 + linhaI_6;
nodo9 = aresta9_10 + aresta9_F + linha9_2 + linha9_10 + linha9_3 + linha9_11 + linha9_4 + linha9_7 + linha9_12 + linha9_F + aresta6_9 + linha6_9 + linha1_9 + linhaI_9;
nodo2 = aresta2_3esq + aresta2_3dir + linha2_10 + linha2_3 + linha2_11 + linha2_4 + linha2_7 + linha2_12 + linha2_F + aresta1_2 + linha9_2 + linha6_2 + linha1_2 + linhaI_2;
```

nodo10 = aresta10_11esq + aresta10_11dir + linha10_3 + linha10_11 + linha10_4 + linha10_7 + linha10_12 + linha10_F + aresta9_10 + linha2_10 + linha9_10 + linha6_10 + linha1_10 + linhaI_10;
nodo3 = aresta3_4 + linha3_11 + linha3_4 + linha3_7 + linha3_12 + linha3_F + aresta2_3esq + aresta2_3dir + linha10_3 + linha2_3 + linha9_3 + linha6_3 + linha1_3 + linhaI_3;
nodo11 = aresta11_12 + linha11_4 + linha11_7 + linha11_12 + linha11_F + aresta10_11esq + aresta10_11dir + linha3_11 + linha10_11 + linha2_11 + linha9_11 + linha6_11 + linha1_11 + linhaI_11;
nodo4 = aresta4_7 + linha4_7 + linha4_12 + linha4_F + aresta3_4 + arestaI_4 + linha11_4 + linha3_4 + linha10_4 + linha2_4 + linha9_4 + linha6_4 + linha1_4 + linhaI_4;
nodo7 = aresta7_12 + linha7_12 + linha7_F + aresta4_7 + linha4_7 + linha11_7 + linha3_7 + linha10_7 + linha2_7 + linha9_7 + linha6_7 + linha1_7 + linhaI_7;
nodo12 = aresta12_F + linha12_F + aresta7_12 + linha7_12 + linha4_12 + linha11_12 + linha3_12 + linha10_12 + linha2_12 + linha9_12 + linha6_12 + linha1_12 + linhaI_12;
nodoF = aresta12_F + aresta9_F + linha12_F + linha7_F + linha4_F + linha11_F + linha3_F + linha10_F + linha2_F + linha9_F + linha6_F + linha1_F + linhaI_F;

//Teste de imparidade - a soma de todos os caminhos que atravessam um nodo tem que
//ser ímpar e maior ou igual a 1.
nodoI = 2 kI + 1;
nodoF = 2 kF + 1;
nodo1 = 2 k1 + 1;
nodo2 = 2 k2 + 1;
nodo3 = 2 k3 + 1;
nodo4 = 2 k4 + 1;
nodo6 = 2 k6 + 1;
nodo7 = 2 k7 + 1;
nodo9 = 2 k9 + 1;
nodo10 = 2 k10 + 1;
nodo11 = 2 k11 + 1;
nodo12 = 2 k12 + 1;
kI >= 0; kF >= 0; k1 >= 0; k2 >= 0; k3 >= 0; k4 >= 0; k6 >= 0; k7 >= 0; k9 >= 0; k10 >= 0; k11 >= 0; k12 >= 0;
int kI, kF, k1, k2, k3, k4, k6, k7, k9, k10, k11, k12;

//Todos as variáveis de decisão (caminhos escolhidos) são valores inteiros, visto que não há "meios" caminhos
int arestaI_1, arestaI_4, linhaI_1, linhaI_6, linhaI_9, linhaI_2, linhaI_10, linhaI_3, linhaI_11;
int linhaI_4, linhaI_7, linhaI_12, linhaI_F, aresta1_6, aresta1_2, linha1_6, linha1_9, linha1_2;
int linha1_10, linha1_3, linha1_11, linha1_4, linha1_7, linha1_12, linha1_F, aresta6_9, aresta6_7;
int linha6_9, linha6_2, linha6_10, linha6_3, linha6_11, linha6_4, linha6_7, linha6_12, linha6_F;
int aresta9_10, aresta9_F, linha9_2, linha9_10, linha9_3, linha9_11, linha9_4, linha9_7, linha9_12;
int linha9_F, aresta2_3esq, aresta2_3dir, linha2_10, linha2_3, linha2_11, linha2_4, linha2_7;
int linha2_12, linha2_F, aresta10_11esq, aresta10_11dir, linha10_3, linha10_11, linha10_4, linha10_7;
int linha10_12, linha10_F, aresta3_4, linha3_11, linha3_4, linha3_7, linha3_12, linha3_F, aresta11_12;
int linha11_4, linha11_7, linha11_12, linha11_F, aresta4_7, linha4_7, linha4_12, linha4_F, aresta7_12;
int linha7_12, linha7_F, aresta12_F, linha12_F;

Ficheiro Output

Variables	MILP	result
	18,24	18,24
custoExtraArestas	0	0
custoExtraLinhas	18,24	18,24
arestaI_1	0	0
arestaI_4	0	0
aresta1_6	0	0
aresta1_2	0	0
aresta6_9	0	0
aresta6_7	0	0
aresta9_10	0	0
aresta9_F	0	0
aresta2_3esq	0	0
aresta2_3dir	0	0
aresta10_11esq	0	0
aresta10_11dir	0	0
aresta3_4	0	0
aresta11_12	0	0
aresta4_7	0	0
aresta7_12	0	0
aresta12_F	0	0
linhaI_1	0	0
linhaI_6	0	0
linhaI_9	0	0
linhaI_2	1	1
linhaI_10	0	0
linhaI_3	0	0
linhaI_11	0	0
linhaI_4	0	0
linhaI_7	0	0
linhaI_12	0	0
linhaI_F	0	0
linha1_6	1	1
linha1_9	0	0
linha1_2	0	0
linha1_10	0	0
linha1_3	0	0
linha1_11	0	0
linha1_4	0	0
linha1_7	0	0
linha1_12	0	0
linha1_F	0	0
linha6_9	0	0
linha6_2	0	0
linha6_10	0	0
linha6_3	0	0
linha6_11	0	0
linha6_4	0	0
linha6_7	0	0
linha6_12	0	0
linha6_F	0	0
linha9_2	0	0
linha9_10	1	1
linha9_3	0	0
linha9_11	0	0
linha9_4	0	0
linha9_7	0	0
linha9_12	0	0
linha9_F	0	0
linha2_10	0	0
linha2_3	0	0
linha2_11	0	0
linha2_4	0	0
linha2_7	0	0
linha2_12	0	0
linha2_F	0	0
linha10_3	0	0
linha10_11	0	0
linha10_4	0	0
linha10_7	0	0
linha10_12	0	0
linha10_F	0	0
linha3_11	0	0
linha3_4	1	1
linha3_7	0	0
linha3_12	0	0
linha3_F	0	0
linha11_4	0	0
linha11_7	1	1
linha11_12	0	0
linha11_F	0	0
linha4_7	0	0
linha4_12	0	0
linha4_F	0	0
linha7_12	0	0
linha7_F	0	0
linha12_F	1	1
nodoI	1	1
nodo1	1	1
nodo6	1	1
nodo9	1	1
nodo2	1	1
nodo10	1	1
nodo3	1	1
nodo11	1	1
nodo4	1	1
nodo7	1	1
nodo12	1	1
nodoF	1	1
kI	0	0
kF	0	0
k1	0	0
k2	0	0
k3	0	0
k4	0	0
k6	0	0
k7	0	0
k9	0	0
k10	0	0
k11	0	0
k12	0	0

*Linhas sombreadas para destaque visual

Solução Ótima

Interpretação do Resultado

Depois de obter os resultados, é necessário recorrer a uma avaliação e análise destes mesmos.

Começamos por denotar o valor correspondente à distância extra do percurso a ser efetuado, ou seja, o valor da função objetivo, tendo obtido um valor de 18.24 U.M. Isto indica que, para além das 80 U.M do percurso obrigatório, o *drone* irá percorrer 18.24 U.M extra de forma que complete o caminho Euleriano, tendo uma distância total de 98.24 U.M.

A próxima observação a ser efetuada será a definição das linhas/arestas extras a serem percorridas no mapa. Para isso, procuramos as variáveis cujo resultado fosse um valor não nulo, tendo verificado os seguintes resultados:

linhaI_2 = 1, linha1_6 = 1, linha9_10 = 1, linha3_4 = 1, linha11_7 = 1, linha12_F = 1

Nota: Linhas como a linha1_6 são equivalentes (em termos de distância) à aresta1_6, no entanto o modelo optou por utilizar as linhas, sendo esta escolha redundante.

De seguida inserimos as linhas obtidas no nosso mapa, para obter uma melhor visualização do caminho a ser efetuado: (**Nota:** As linhas curvas representam a duplicação do caminho da aresta entre os vértices, estando apenas curvas de forma a facilitar a visualização.)

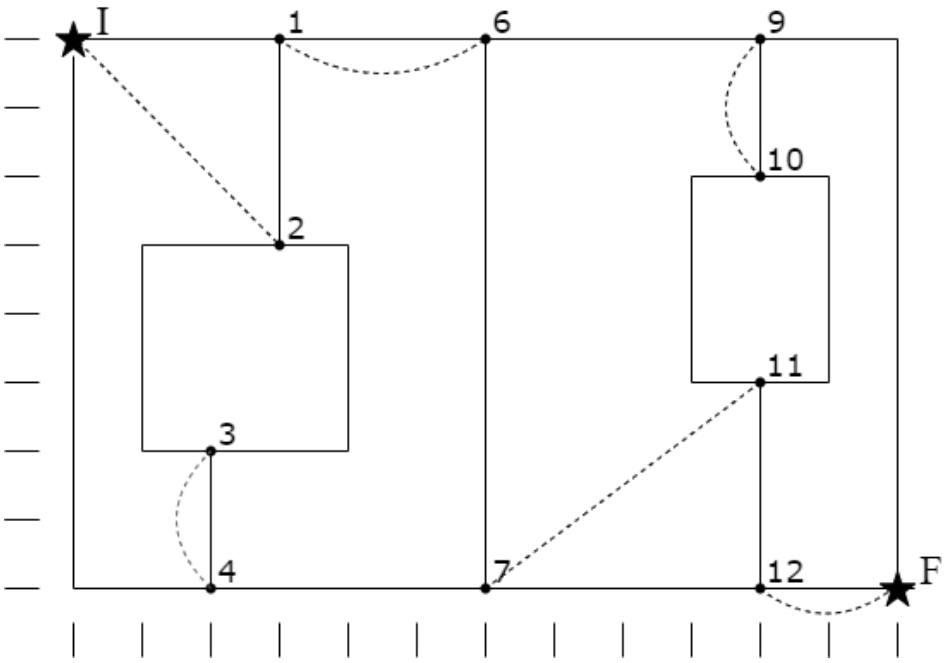

Figura 2 - Mapa do Caminho Euleriano

Por último, avaliamos os valores obtidos nas variáveis "nodos" e os respetivos valores de k. Estes possuem os seus valores mínimos (nodos = 1 e k = 0), ou seja, o modelo não adicionou mais de uma aresta por nodo e todos os nodos foram visitados, tal como requerido.

Definição do Percurso

Depois dos resultados analisados e a solução ótima obtida, finalizamos o problema efetuando o cálculo do percurso que o *drone* terá de tomar. Decidimos que este iria tomar preferencialmente o caminho mais à direita, obtendo o seguinte percurso final:

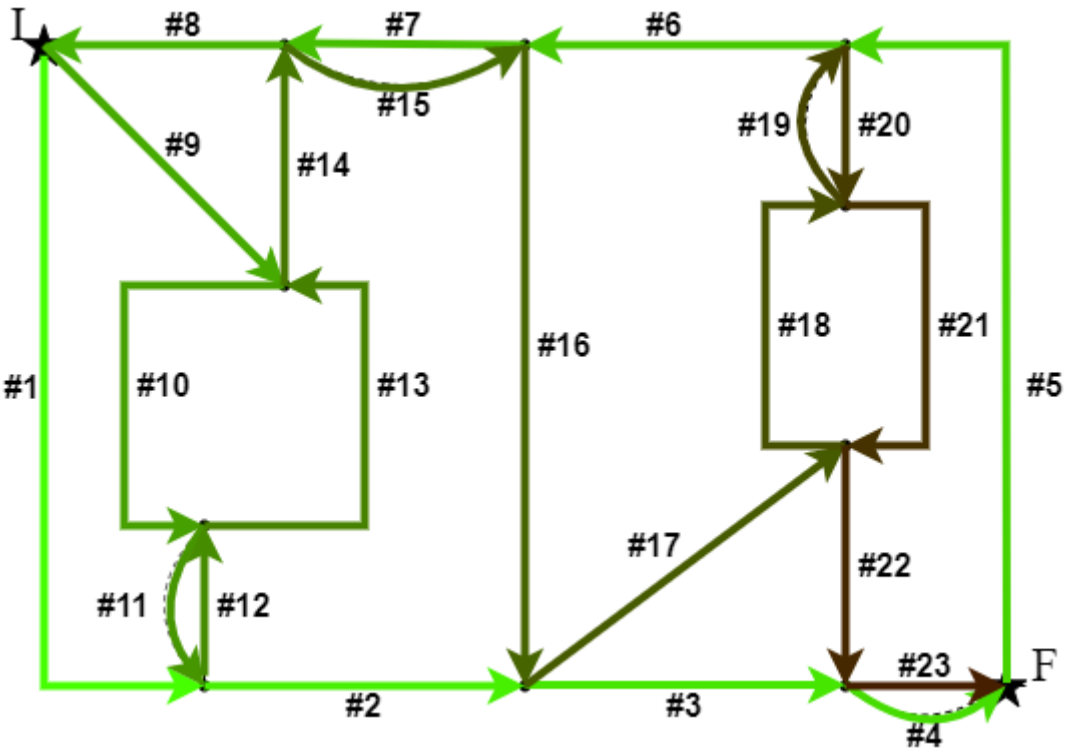

Figura 3 - Percurso do Drone

Ordem dos Nodos percorridos:

I -> 4 -> 7 -> 2 -> F -> 9 -> 6 -> 1 -> I -> 2 -> 3 -> 4 -> 3 -> 2 -> 1 -> 6 -> 7 -> 11 -> 10 -> 9 -> 10 -> 11 -> 12 -> F

Custo Associado a cada Caminho:

10 -> 4 -> 4 -> 2 -> 10 -> 4 -> 3 -> 3 -> 4.24 -> 6 -> 2 -> 2 -> 6 -> 3 -> 3-> 8 -> 5 -> 5 -> 2 -> 2 -> 5 -> 3 -> 2

Custo Associado Acumulado:

10 -> 14 -> 18 -> 20 -> 30 -> 34 -> 37 -> 40 -> 44.24 -> 50.24 -> 52.24 -> 54.24 -> 60.24 -> 63.24 -> 66.24 -> 74.24 -> 79.24 -> 84.24 -> 86.24 -> 88.24 -> 93.24 -> 96.24 -> 98.24

Assim obtemos o percurso final do drone, percorrendo todas as arestas obrigatórias e terminando no nodo requerido, com um custo de distância total de 98.25 U.M.

Validação do Modelo

Para validar o modelo, decidimos alterar o problema ligeiramente, de forma a testar a adaptabilidade do mesmo. Decidimos alterar o enunciando apenas na restrição do nodo de partida e chegada do *drone*, alterando esta para que o *drone* comece no nodo I e termine a inspeção no nodo I. Assim, teremos um problema de circuito Euleriano.

No modelo, apenas alteramos as seguintes restrições: NodoI passa a ter a restrição $2 \times kI, kI \in \mathbb{N}_0$ e o NodoF muda a sua restrição para $2 \times kF, kF \in \mathbb{N}_0$. Estas alterações foram necessárias visto que, num circuito Euleriano, é necessário que todos os nodos contenham um número par de arestas.

Depois destas mudanças, voltamos a executar o modelo, obtendo este mapa final:

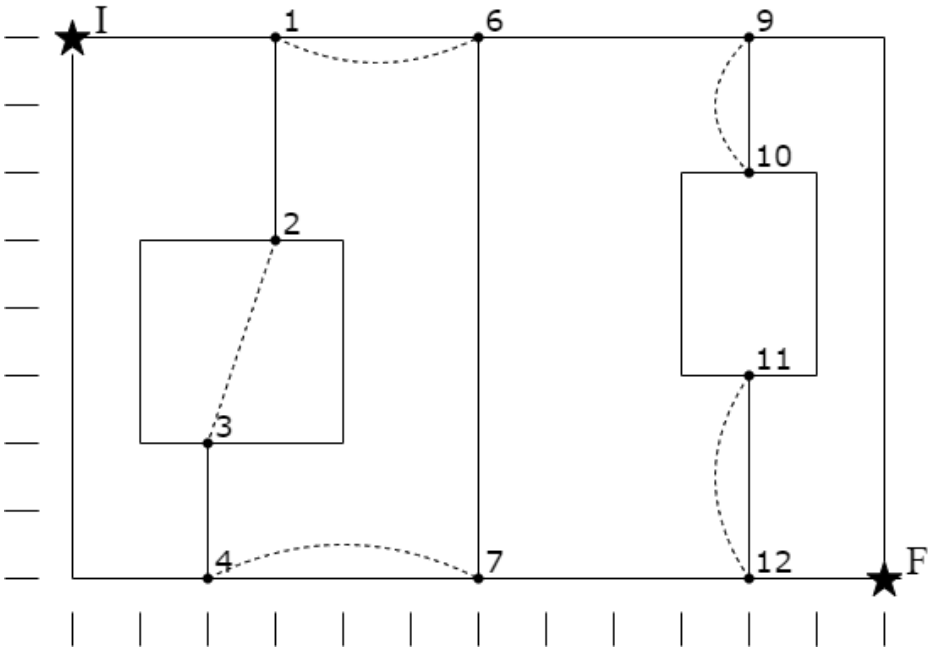

Figura 4 - Mapa do Circuito Euleriano

Este mapa satisfaz as condições para um circuito Euleriano, tendo uma distância extra percorrida de 15,16 U.M, e uma distância total de 95,16 U.M. Concluímos assim que o modelo desenvolvido é capaz de se adaptar às novas restrições, provando assim a sua validade.

Conclusão

Com a realização deste trabalho prático, acreditamos que alcançamos com sucesso uma das possíveis soluções ótimas do problema que nos foi proposto. Também acreditamos ter aprofundado o nosso conhecimento na área de otimização de caminhos *Eulerianos* assim como na interpretação, formulação e desenvolvimento de problemas de otimização.