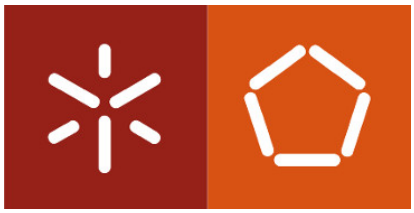


Universidade do Minho

Mestrado Integrado em Engenharia Informática

Investigação Operacional - Trabalho Prático #1
Ano Letivo 2021/2022

Gonçalo Braz (a93178) Simão Cunha (a93262)
Tiago Silva (a93277) Gonçalo Pereira (a93168)

24 de março de 2022

1 Formulação do problema

O problema que nos foi colocado para este primeiro trabalho prático da UC de Investigação Operacional consiste num veículo não tripulado (*drone*) inspecionar linhas de transporte de energia elétrica em alta tensão para verificar se há vegetação a interferir com as linhas. Foi imposto que o *drone* possa percorrer as arestas nos dois sentidos e pode percorrer mais do que uma vez, não sendo necessariamente que viaje pelas linhas - pode viajar pelo ar. Além disso, tem-se como objetivo minimizar esta distância. Iremos utilizar o *software* LPSolve para a resolução deste problema.

Tendo que conta que o maior número de aluno existente no nosso grupo (**93277**), devemos remover a aresta C, surgindo, assim, o seguinte mapa:

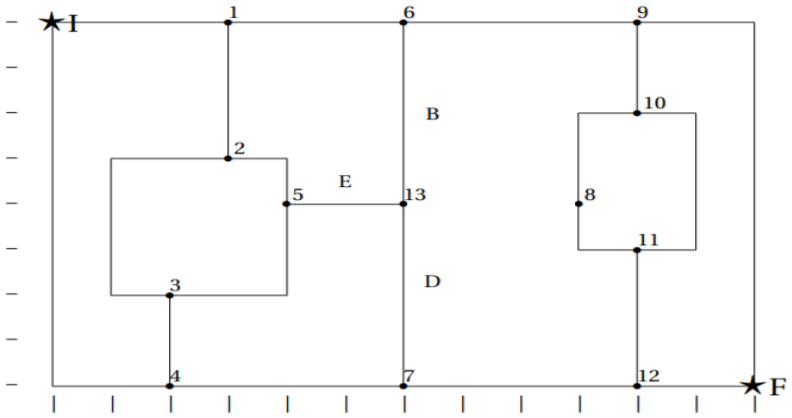


Figura 1: Mapa resultante da remoção da aresta C

Além de ter de percorrer todas as linhas, o *drone* tem definido um ponto de início e fim de operação. Estamos, portanto, perante um problema de resolução de um *caminho Euleriano*. No entanto, para podermos resolver este problema, para além de o grafo ter de ser ligado, necessitamos de saber se estamos a trabalhar com um grafo que possua, no máximo, dois vértices de grau ímpar, tal como dito na definição 5 do capítulo 5 de *Modelos de Investigação Operacional* (livro de apontamentos da UC).

Analisando o grafo que nos foi proposto, facilmente verificamos que este não pode apresentar um caminho Euleriano, visto que apresenta mais do que dois vértices de grau ímpar: ex. vértice 1, 6, 9, etc.

Deste modo teremos de modificar o nosso grafo de modo a que todos os seus vértices tenham grau par ou tenham no máximo dois vértices de grau ímpar.

Uma solução simples será considerar o grafo como orientado e, para que o *drone* possa atravessar em qualquer dos sentidos, dobramos o número de arestas - um exemplo disto será considerar que, em vez de termos apenas a aresta *i_1*, temos o seu par *1_i*. Tendo isto em conta, todos os vértices terão o seu grau de arestas dobrado, isto é, todos passarão a ter grau par, podendo então concluir que será possível a existência de um caminho Euleriano dentro deste.

Adicionalmente, no contexto do nosso problema, visto que o sujeito que irá atravessar as linhas de alta tensão é um *drone*, este poderá movimentar-se no ar entre os vértices, não sendo limitado por estas linhas, ou seja, podemos adicionar arestas entre quaisquer vértices.

Assim sendo, tendo em conta que o grafo possuí 15 vértices, podemos conectar qualquer vértice aos 14 restantes, usando claro a mesma lógica anterior sobre a orientação de forma a manter o grau dos vértices pares. Consequentemente, permitimos ao *drone* seguir caminhos mais diretos entre os vértices no caso de ter de retroceder, como é exemplo a ligação entre i e 4: pela linha elétrica terá custo 10, porém seguindo diretamente terá apenas custo de 8.25.

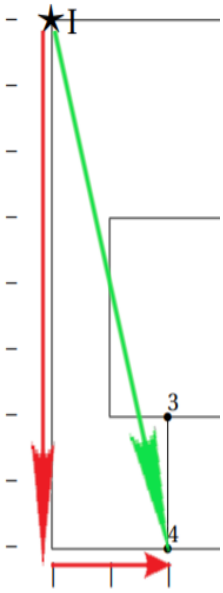


Figura 2: Distância euclidiana de I para 4

O valor das distâncias das arestas seguindo a linha é facilmente descoberto seguindo a escala de traços nas extremidades do mapa, e estará apresentado na função objetivo no capitulo do modelo adiante.

Quanto às distâncias euclidianas entre os vértices, das quais as arestas aéreas que consideramos irão dispor, foi-nos fornecida por parte dos docentes a seguinte matriz:

x	0	3	3	2	2	4	6	6	9	10	10	10	10	6	12
y	8	8	5	2	0	4	8	0	4	8	6	3	0	4	0

x	y		I	1	2	3	4	5	6	7	8	9	10	11	12	13	F
0	8	I	0,00	3,00	4,24	6,32	8,25	5,66	6,00	10,00	9,85	10,00	10,20	11,18	12,81	7,21	14,42
3	8	1		0,00	3,00	6,08	8,06	4,12	3,00	8,54	7,21	7,00	7,28	8,60	10,63	5,00	12,04
3	5	2			0,00	3,16	5,10	1,41	4,24	5,83	6,08	7,62	7,07	7,28	8,60	3,16	10,30
2	2	3				0,00	2,00	2,83	7,21	4,47	7,28	10,00	8,94	8,06	8,25	4,47	10,20
2	0	4					0,00	4,47	8,94	4,00	8,06	11,31	10,00	8,54	8,00	5,66	10,00
4	4	5						0,00	4,47	4,47	5,00	7,21	6,32	6,08	7,21	2,00	8,94
6	8	6							0,00	8,00	5,00	4,00	4,47	6,40	8,94	4,00	10,00
6	0	7								0,00	5,00	8,94	7,21	5,00	4,00	4,00	6,00
9	4	8									0,00	4,12	2,24	1,41	4,12	3,00	5,00
10	8	9										0,00	2,00	5,00	8,00	5,66	8,25
10	6	10											0,00	3,00	6,00	4,47	6,32
10	3	11												0,00	3,00	4,12	3,61
10	0	12													0,00	5,66	2,00
6	4	13														0,00	7,21
12	0	F															0,00

Figura 3: Distâncias euclidianas entre todos os vértices

Com todas estas considerações em mente, estamos prontos para procurar a solução do caminho euleriano das linhas de tensão percorridas pelo *drone*.

2 Modelo

A construção do modelo de programação linear consiste na formulação de expressões para a função objetivo e para as restrições do problema, tendo em conta as variáveis de decisão em causa. Iremos dividir a construção do nosso modelo em 3 secções:

1. `Variáveis de decisão`: irão ser as incógnitas, traduzindo o facto se o drone passa por uma determinada linha de alta tensão (ou por via aérea) e quantas vezes passa por essa aresta;
2. `Função objetivo`: responsável por definir o problema consoante as variáveis de função. Determina a melhor solução (de entre as soluções válidas).
3. `Restrições`: representam as condições associadas ao nosso problema. Uma solução que respeite estas restrições é considerada válida;

2.1 Variáveis de decisão

O primeiro passo é saber como iremos representar as diversas arestas. Decidimos que iremos representar as arestas no formato:

$$x_{i\_j} \tag{1}$$

onde i representa o vértice de origem e j o vértice de destino.

Tendo em conta que temos de ter em atenção a arestas aéreas, iremos representar as ligações onde existe uma linha de alta tensão entre dois vértices (mas existe a possibilidade de efetuar uma ligação que representa a distância euclidiana) através do formato:

$$x_{i\_j\_A} \tag{2}$$

Por exemplo, para representar uma ligação entre o vértice i e *12*, representaremos por x_{i_12}, enquanto que uma ligação entre o vértice *12* e j será representada por x_{12_i}

2.2 Função objetivo

A função objetivo dependerá do número de vezes que o *drone* percorre cada linha, minimizando esta função de modo a obter o menor custo possível. Com esta ideia, sabemos que estamos perante um problema de programação linear de minimização.

Assim, a nossa função objetivo seguirá o formato:

$$z = \sum c_{i\_j} x_{i\_j} \tag{3}$$

onde c_{i_j} representa o custo de ir do vértice. Entenda-se *custo* como a distância entre dois vértices. De acordo com o nosso problema, temos de incluir:

- todas as arestas "reais"(i.e. que existem efetivamente no grafo). O custo irá ser representado pela distância euclidiana entre os dois vértices constituintes. **Exemplo**: 3.00 x_{i_1}
- as arestas *aéreas* (i.e. que não existem desenhadas no grafo). O custo irá ser representado pela distância euclidiana entre os dois vértices constituintes. **Exemplo**: 4.24 x_{i_2}
- as arestas aéreas que representam as ligações em linha reta entre vértices ligados por um fio de alta tensão. O custo irá ser representado pelo menor comprimento das linhas que formam a ligação entre os dois vértices constituintes. **Exemplo**: 8.25 $x_{i\_4\_A}$

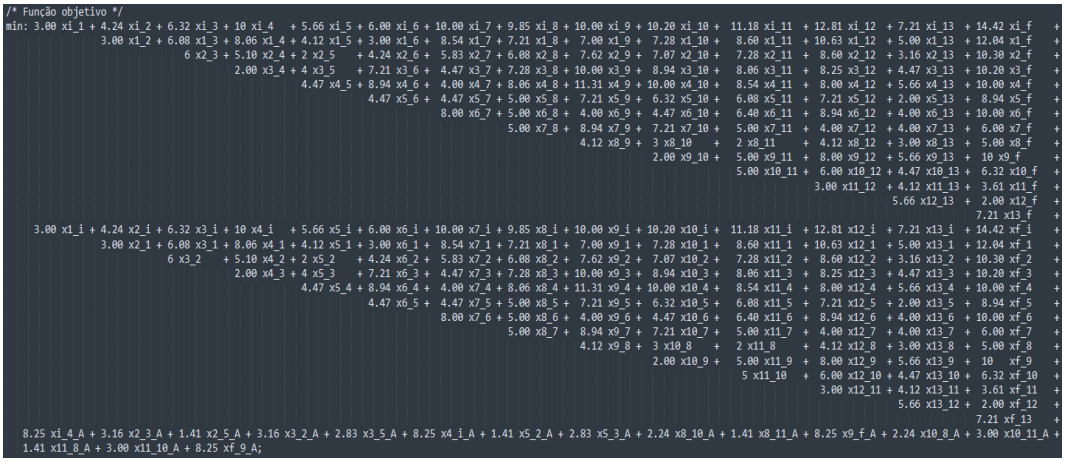
2.3 Restrições

As restrições do modelo representam as condições para que uma solução seja considerada válida. Assim, dividimos as restrições em 4 conjuntos:

- As linhas de alta tensão têm de ser percorridas pelo *drone* pelo menos uma vez, não importando o sentido: $x_{i\_j} + x_{j\_i} >= 1$
- As linhas poderão ser percorridas (incluindo as aéreas). Para esta restrição, temos de incluir todas as arestas possíveis. Corresponderá ao formato $x_{i\_j} >= 0$.
- O número de arestas que entram num vértice tem de ser o mesmo número de arestas que saem do mesmo: $\sum x_{i\_j} = \sum x_{j\_i}$. Esta regra não se aplica para os vértices I e F visto que estes são os nodos final e inicial. No caso do nodo I, por ser o inicial, terão de sair mais arestas dele do que os que entram, mais especificamente, menos uma aresta entra do que os que sai e o contrário para o F.

2.4 Ficheiro de input inserido no `LPSolve`

De forma a ser possível observar o conteúdo do ficheiro de input inserido no `LPSolve` e tendo em conta que não é possível fazer *zoom-out*, tivemos que abrir o mesmo com recurso ao editor de texto `Sublime Text`.



```
/* Função objetivo */
min: 3.00 xi_1 + 4.24 xi_2 + 6.32 xi_3 + 10 xi_4   + 5.66 xi_5 + 6.00 xi_6 + 10.00 xi_7 + 9.85 xi_8 + 10.00 xi_9 + 10.20 xi_10 +  11.18 xi_11  + 12.81 xi_12  + 7.21 xi_13  + 14.42 xi_f    +
              3.00 x1_2 + 6.08 x1_3 + 8.06 x1_4 + 4.12 x1_5 + 3.00 x1_6 +  8.54 x1_7 + 7.21 x1_8 +  7.00 x1_9 +  7.28 x1_10 +   8.60 x1_11  + 10.63 x1_12  + 5.00 x1_13  + 12.04 x1_f    +
                           6 x2_3 + 5.10 x2_4 + 2 x2_5    + 4.24 x2_6 +  5.83 x2_7 + 6.08 x2_8 +  7.62 x2_9 +  7.07 x2_10 +   7.28 x2_11  +  8.60 x2_12  + 3.16 x2_13  + 10.30 x2_f    +
                                      2.00 x3_4 + 4 x3_5    + 7.21 x3_6 +  4.47 x3_7 + 7.28 x3_8 + 10.00 x3_9 +  8.94 x3_10 +   8.06 x3_11  +  8.25 x3_12  + 4.47 x3_13  + 10.20 x3_f    +
                                                 4.47 x4_5 + 8.94 x4_6 +  4.00 x4_7 + 8.06 x4_8 + 11.31 x4_9 + 10.00 x4_10 +   8.54 x4_11  +  8.00 x4_12  + 5.66 x4_13  + 10.00 x4_f    +
                                                             4.47 x5_6 +  4.47 x5_7 + 5.00 x5_8 +  7.21 x5_9 +  6.32 x5_10 +   6.08 x5_11  +  7.21 x5_12  + 2.00 x5_13  +  8.94 x5_f    +
                                                                         8.00 x6_7 + 5.00 x6_8 +  4.00 x6_9 +  4.47 x6_10 +   6.40 x6_11  +  8.94 x6_12  + 4.00 x6_13  + 10.00 x6_f    +
                                                                                     5.00 x7_8 +  8.94 x7_9 +  7.21 x7_10 +   5.00 x7_11  +  4.00 x7_12  + 4.00 x7_13  +  6.00 x7_f    +
                                                                                                 4.12 x8_9 +  3 x8_10    +   2 x8_11    +  4.12 x8_12  + 3.00 x8_13  +  5.00 x8_f    +
                                                                                                              2.00 x9_10 +   5.00 x9_11  +  8.00 x9_12  + 5.66 x9_13  +  10 x9_f     +
                                                                                                                             5.00 x10_11 +  6.00 x10_12 + 4.47 x10_13 +  6.32 x10_f    +
                                                                                                                                          3.00 x11_12  + 4.12 x11_13 +  3.61 x11_f    +
                                                                                                                                                         5.66 x12_13  +  2.00 x12_f    +
                                                                                                                                                                         7.21 x13_f    +
    3.00 x1_i + 4.24 x2_i + 6.32 x3_i + 10 x4_i   + 5.66 x5_i + 6.00 x6_i + 10.00 x7_i + 9.85 x8_i + 10.00 x9_i + 10.20 x10_i +  11.18 x11_i  + 12.81 x12_i  + 7.21 x13_i  + 14.42 xf_i    +
              3.00 x2_1 + 6.08 x3_1 + 8.06 x4_1 + 4.12 x5_1 + 3.00 x6_1 +  8.54 x7_1 + 7.21 x8_1 +  7.00 x9_1 +  7.28 x10_1 +   8.60 x11_1  + 10.63 x12_1  + 5.00 x13_1  + 12.04 xf_1    +
                         6 x3_2    + 5.10 x4_2 + 2 x5_2    + 4.24 x6_2 +  5.83 x7_2 + 6.08 x8_2 +  7.62 x9_2 +  7.07 x10_2 +   7.28 x11_2  +  8.60 x12_2  + 3.16 x13_2  + 10.30 xf_2    +
                                      2.00 x4_3 + 4 x5_3    + 7.21 x6_3 +  4.47 x7_3 + 7.28 x8_3 + 10.00 x9_3 +  8.94 x10_3 +   8.06 x11_3  +  8.25 x12_3  + 4.47 x13_3  + 10.20 xf_3    +
                                                 4.47 x5_4 + 8.94 x6_4 +  4.00 x7_4 + 8.06 x8_4 + 11.31 x9_4 + 10.00 x10_4 +   8.54 x11_4  +  8.00 x12_4  + 5.66 x13_4  + 10.00 xf_4    +
                                                             4.47 x6_5 +  4.47 x7_5 + 5.00 x8_5 +  7.21 x9_5 +  6.32 x10_5 +   6.08 x11_5  +  7.21 x12_5  + 2.00 x13_5  +  8.94 xf_5    +
                                                                         8.00 x7_6 + 5.00 x8_6 +  4.00 x9_6 +  4.47 x10_6 +   6.40 x11_6  +  8.94 x12_6  + 4.00 x13_6  + 10.00 xf_6    +
                                                                                     5.00 x8_7 +  8.94 x9_7 +  7.21 x10_7 +   5.00 x11_7  +  4.00 x12_7  + 4.00 x13_7  +  6.00 xf_7    +
                                                                                                 4.12 x9_8 +  3 x10_8    +   2 x11_8    +  4.12 x12_8  + 3.00 x13_8  +  5.00 xf_8    +
                                                                                                              2.00 x10_9 +   5.00 x11_9  +  8.00 x12_9  + 5.66 x13_9  +  10   xf_9    +
                                                                                                                              5 x11_10   +  6.00 x12_10 + 4.47 x13_10 +  6.32 xf_10   +
                                                                                                                                          3.00 x12_11 + 4.12 x13_11 +  3.61 xf_11   +
                                                                                                                                                         5.66 x13_12 +  2.00 xf_12   +
                                                                                                                                                                        7.21 xf_13   +
  8.25 xi_4_A + 3.16 x2_3_A + 1.41 x2_5_A + 3.16 x3_2_A + 2.83 x3_5_A + 8.25 x4_i_A + 1.41 x5_2_A + 2.83 x5_3_A + 2.24 x8_10_A + 1.41 x8_11_A + 8.25 x9_f_A + 2.24 x10_8_A + 3.00 x10_11_A +
  1.41 x11_8_A + 3.00 x11_10_A + 8.25 xf_9_A;
```

Figura 4: Função objetivo

```
/* Restrições */

// As arestas que existem no grafo têm de ser percorridas pelo menos uma vez (não importa o sentido)

xi_1 + x1_i >= 1; xi_4 + x4_i >= 1;
x1_2 + x2_1 >= 1; x1_6 + x6_1 >= 1;
x2_3 + x3_2 >= 1; x2_5 + x5_2 >= 1;
x3_4 + x4_3 >= 1; x3_5 + x5_3 >= 1;
x4_7 + x7_4 >= 1;
x5_13 + x13_5 >= 1;
x6_13 + x13_6 >= 1; x6_9 + x9_6 >= 1;
x7_13 + x13_7 >= 1; x7_12 + x12_7 >= 1;
x8_10 + x10_8 >= 1; x8_11 + x11_8 >= 1;
x10_11 + x11_10 >= 1;
x9_10 + x10_9 >= 1; x9_f + xf_9 >= 1;
x11_12 + x12_11 >= 1; x12_f + xf_12 >= 1;
```

Figura 5: Restrição 1

```
// As arestas poderão ser percorridas (incluindo as aéreas)

restricoesArestasi:   xi_1 >= 0;  xi_2 >= 0;  xi_3 >= 0;  xi_4 >= 0;   xi_5 >= 0;   xi_6>= 0;   xi_7 >= 0;   xi_8 >= 0;   xi_9 >= 0;   xi_10 >= 0;  xi_11 >= 0;  xi_12 >= 0;   xi_13 >= 0;  xi_f  >= 0;
                      xi_4_A >= 0;
restricoesArestas1:   x1_i >= 0;  x1_2 >= 0;  x1_3 >= 0;  x1_4 >= 0;   x1_5 >= 0;   x1_6>= 0;   x1_7 >= 0;   x1_8 >= 0;   x1_9 >= 0;   x1_10 >= 0;  x1_11 >= 0;  x1_12 >= 0;   x1_13 >= 0;  x1_f  >= 0;
restricoesArestas2:   x2_1 >= 0;  x2_i >= 0;  x2_3 >= 0;  x2_4 >= 0;   x2_5 >= 0;   x2_6>= 0;   x2_7 >= 0;   x2_8 >= 0;   x2_9 >= 0;   x2_10 >= 0;  x2_11 >= 0;  x2_12 >= 0;   x2_13 >= 0;  x2_f  >= 0;
                      x2_3_A >= 0; x2_5_A >= 0;
restricoesArestas3:   x3_1 >= 0;  x3_2 >= 0;  x3_i >= 0;  x3_4 >= 0;   x3_5 >= 0;   x3_6>= 0;   x3_7 >= 0;   x3_8 >= 0;   x3_9 >= 0;   x3_10 >= 0;  x3_11 >= 0;  x3_12 >= 0;   x3_13 >= 0;  x3_f  >= 0;
                      x3_2_A >= 0; x3_5_A >= 0;
restricoesArestas4:   x4_1 >= 0;  x4_2 >= 0;  x4_3 >= 0;  x4_i >= 0;   x4_5 >= 0;   x4_6>= 0;   x4_7 >= 0;   x4_8 >= 0;   x4_9 >= 0;   x4_10 >= 0;  x4_11 >= 0;  x4_12 >= 0;   x4_13 >= 0;  x4_f  >= 0;
                      x4_i_A >= 0;
restricoesArestas5:   x5_1 >= 0;  x5_2 >= 0;  x5_3 >= 0;  x5_4 >= 0;   x5_i >= 0;   x5_6>= 0;   x5_7 >= 0;   x5_8 >= 0;   x5_9 >= 0;   x5_10 >= 0;  x5_11 >= 0;  x5_12 >= 0;   x5_13 >= 0;  x5_f  >= 0;
                      x5_2_A >= 0; x5_3_A >= 0;
restricoesArestas6:   x6_1 >= 0;  x6_2 >= 0;  x6_3 >= 0;  x6_4 >= 0;   x6_5 >= 0;   x6_i>= 0;   x6_7 >= 0;   x6_8 >= 0;   x6_9 >= 0;   x6_10 >= 0;  x6_11 >= 0;  x6_12 >= 0;   x6_13 >= 0;  x6_f  >= 0;
restricoesArestas7:   x7_1 >= 0;  x7_2 >= 0;  x7_3 >= 0;  x7_4 >= 0;   x7_5 >= 0;   x7_6>= 0;   x7_i >= 0;   x7_8 >= 0;   x7_9 >= 0;   x7_10 >= 0;  x7_11 >= 0;  x7_12 >= 0;   x7_13 >= 0;  x7_f  >= 0;
restricoesArestas8:   x8_1 >= 0;  x8_2 >= 0;  x8_3 >= 0;  x8_4 >= 0;   x8_5 >= 0;   x8_6>= 0;   x8_7 >= 0;   x8_i >= 0;   x8_9 >= 0;   x8_10 >= 0;  x8_11 >= 0;  x8_12 >= 0;   x8_13 >= 0;  x8_f  >= 0;
                      x8_10_A >= 0; x8_11_A >= 0;
restricoesArestas9:   x9_1 >= 0;  x9_2 >= 0;  x9_3 >= 0;  x9_4 >= 0;   x9_5 >= 0;   x9_6>= 0;   x9_7 >= 0;   x9_8 >= 0;   x9_i >= 0;   x9_10 >= 0;  x9_11 >= 0;  x9_12 >= 0;   x9_13 >= 0;  x9_f  >= 0;
                      x9_f_A >= 0;
restricoesArestas10: x10_1 >= 0; x10_2 >= 0; x10_3 >= 0; x10_4 >= 0;  x10_5 >= 0;  x10_6>= 0;  x10_7 >= 0;  x10_8 >= 0;  x10_9 >= 0;  x10_i  >= 0; x10_11 >= 0; x10_12 >= 0;  x10_13 >= 0; x10_f  >= 0;
                      x10_8_A >= 0; x10_11_A >= 0;
restricoesArestas11: x11_1 >= 0; x11_2 >= 0; x11_3 >= 0; x11_4 >= 0;  x11_5 >= 0;  x11_6>= 0;  x11_7 >= 0;  x11_8 >= 0;  x11_9 >= 0;  x11_10 >= 0;  x11_i >= 0; x11_12 >= 0;  x11_13 >= 0; x11_f  >= 0;
                      x11_8_A >= 0; x11_10_A >= 0;
restricoesArestas12: x12_1 >= 0; x12_2 >= 0; x12_3 >= 0; x12_4 >= 0;  x12_5 >= 0;  x12_6>= 0;  x12_7 >= 0;  x12_8 >= 0;  x12_9 >= 0;  x12_10 >= 0; x12_11 >= 0;  x12_i >= 0;  x12_13 >= 0; x12_f  >= 0;
restricoesArestas13: x13_1 >= 0; x13_2 >= 0; x13_3 >= 0; x13_4 >= 0;  x13_5 >= 0;  x13_6>= 0;  x13_7 >= 0;  x13_8 >= 0;  x13_9 >= 0;  x13_10 >= 0; x13_11 >= 0; x13_12 >= 0;   x13_i >= 0; x13_f  >= 0;
restricoesArestasf:   xf_1 >= 0;  xf_2 >= 0;  xf_3 >= 0;  xf_4 >= 0;   xf_5 >= 0;   xf_6>= 0;   xf_7 >= 0;   xf_8 >= 0;   xf_9 >= 0;   xf_10 >= 0   xf_11 >= 0;  xf_12 >= 0;   xf_13 >= 0;  xf_i  >= 0;
                      xf_9_A >= 0;
```

Figura 6: Restrição 2

```
// O nº de arestas que entram num vértice tem de ser o mesmo nº de arestas que saem

fluxoVertice1:  xi_1 + xi_2 + xi_3 + xi_4 + xi_5 + xi_6 + xi_7 + xi_8 + xi_9 + xi_10 + xi_11 + xi_12 + xi_13 + xi_f + xi_4_A -1 =
                 x1_i + x2_i + x3_i + x4_i + x5_i + x6_i + x7_i + x8_i + x9_i + x10_i + x11_i + x12_i + x13_i + xf_i + x4_i_A;

fluxoVertice1:  x1_i + x1_2 + x1_3 + x1_4 + x1_5 + x1_6 + x1_7 + x1_8 + x1_9 + x1_10 + x1_11 + x1_12 + x1_13 + x1_f =
                 xi_1 + x2_1 + x3_1 + x4_1 + x5_1 + x6_1 + x7_1 + x8_1 + x9_1 + x10_1 + x11_1 + x12_1 + x13_1 + xf_1;

fluxoVertice2:  x2_1 + x2_i + x2_3 + x2_4 + x2_5 + x2_6 + x2_7 + x2_8 + x2_9 + x2_10 + x2_11 + x2_12 + x2_13 + x2_f + x2_3_A + x2_5_A=
                 xi_2 + x1_2 + x3_2 + x4_2 + x5_2 + x6_2 + x7_2 + x8_2 + x9_2 + x10_2 + x11_2 + x12_2 + x13_2 + xf_2 + x3_2_A + x5_2_A;

fluxoVertice3:  x3_1 + x3_2 + x3_i + x3_4 + x3_5 + x3_6 + x3_7 + x3_8 + x3_9 + x3_10 + x3_11 + x3_12 + x3_13 + x3_f + x3_2_A + x3_5_A =
                 x1_3 + x2_3 + xi_3 + x4_3 + x5_3 + x6_3 + x7_3 + x8_3 + x9_3 + x10_3 + x11_3 + x12_3 + x13_3 + xf_3 + x2_3_A + x5_3_A;

fluxoVertice4:  x4_1 + x4_2 + x4_3 + x4_i + x4_5 + x4_6 + x4_7 + x4_8 + x4_9 + x4_10 + x4_11 + x4_12 + x4_13 + x4_f + x4_i_A =
                 x1_4 + x2_4 + x3_4 + xi_4 + x5_4 + x6_4 + x7_4 + x8_4 + x9_4 + x10_4 + x11_4 + x12_4 + x13_4 + xf_4 + xi_4_A;

fluxoVertice5:  x5_1 + x5_2 + x5_3 + x5_4 + x5_i + x5_6 + x5_7 + x5_8 + x5_9 + x5_10 + x5_11 + x5_12 + x5_13 + x5_f + x5_2_A + x5_3_A=
                 x1_5 + x2_5 + x3_5 + x4_5 + xi_5 + x6_5 + x7_5 + x8_5 + x9_5 + x10_5 + x11_5 + x12_5 + x13_5 + xf_5 + x2_5_A + x3_5_A;

fluxoVertice6:  x6_1 + x6_2 + x6_3 + x6_4 + x6_5 + x6_i + x6_7 + x6_8 + x6_9 + x6_10 + x6_11 + x6_12 + x6_13 + x6_f =
                 x1_6 + x2_6 + x3_6 + x4_6 + x5_6 + xi_6 + x7_6 + x8_6 + x9_6 + x10_6 + x11_6 + x12_6 + x13_6 + xf_6;

fluxoVertice7:  x7_1 + x7_2 + x7_3 + x7_4 + x7_5 + x7_6 + x7_i + x7_8 + x7_9 + x7_10 + x7_11 + x7_12 + x7_13 + x7_f =
                 xi_7 + x2_7 + x3_7 + x4_7 + x5_7 + x6_7 + xi_7 + x8_7 + x9_7 + x10_7 + x11_7 + x12_7 + x13_7 + xf_7;

fluxoVertice8:  x8_1 + x8_2 + x8_3 + x8_4 + x8_5 + x8_6 + x8_7 + x8_i + x8_9 + x8_10 + x8_11 + x8_12 + x8_13 + x8_f + x8_10_A + x8_11_A=
                 x1_8 + x2_8 + x3_8 + x4_8 + x5_8 + x6_8 + x7_8 + xi_8 + x9_8 + x10_8 + x11_8 + x12_8 + x13_8 + xf_8 + x10_8_A + x11_8_A;

fluxoVertice9:  x9_1 + x9_2 + x9_3 + x9_4 + x9_5 + x9_6 + x9_7 + x9_8 + x9_i + x9_10 + x9_11 + x9_12 + x9_13 + x9_f + x9_f_A=
                 x1_9 + x2_9 + x3_9 + x4_9 + x5_9 + x6_9 + x7_9 + x8_9 + xi_9 + x10_9 + x11_9 + x12_9 + x13_9 + xf_9 + xf_9_A;

fluxoVertice10: x10_1 + x10_2 + x10_3 + x10_4 + x10_5 + x10_6 + x10_7 + x10_8 + x10_9 + x10_i + x10_11 + x10_12 + x10_13 + x10_f + x10_8_A + x10_11_A=
                 x1_10 + x2_10 + x3_10 + x4_10 + x5_10 + x6_10 + x7_10 + x8_10 + x9_10 + xi_10 + x11_10 + x12_10 + x13_10 + xf_10 + x8_10_A + x11_10_A;

fluxoVertice11: x11_1 + x11_2 + x11_3 + x11_4 + x11_5 + x11_6 + x11_7 + x11_8 + x11_9 + x11_10 + x11_i + x11_12 + x11_13 + x11_f + x11_8_A + x11_10_A=
                 x1_11 + x2_11 + x3_11 + x4_11 + x5_11 + x6_11 + x7_11 + x8_11 + x9_11 + x10_11 + xi_11 + x12_11 + x13_11 + xf_11 + x8_11_A +x10_11_A;

fluxoVertice12: x12_1 + x12_2 + x12_3 + x12_4 + x12_5 + x12_6 + x12_7 + x12_8 + x12_9 + x12_10 + x12_11 + x12_i + x12_13 + x12_f =
                 x1_12 + x2_12 + x3_12 + x4_12 + x5_12 + x6_12 + x7_12 + x8_12 + x9_12 + x10_12 + x11_12 + xi_12 + x13_12 + xf_12;

fluxoVertice13: x13_1 + x13_2 + x13_3 + x13_4 + x13_5 + x13_6 + x13_7 + x13_8 + x13_9 + x13_10 + x13_11 + x13_12 + x13_i + x13_f =
                 x1_13 + x2_13 + x3_13 + x4_13 + x5_13 + x6_13 + x7_13 + x8_13 + x9_13 + x10_13 + x11_13 + x12_13 + xi_13 + xf_13;

fluxoVerticef:  xf_1 + xf_2 + xf_3 + xf_4 + xf_5 + xf_6 + xf_7 + xf_8 + xf_9 + xf_10 + xf_11 + xf_12 + xf_13 + xf_i + xf_9_A=
                 x1_f + x2_f + x3_f + x4_f + x5_f + x6_f + x7_f + x8_f + x9_f + x10_f + x11_f + x12_f + x13_f + xi_f + x9_f_A -1;
```

Figura 7: Restrição 3

```
// Tipo das variáveis

int  xi_1,  xi_2,  xi_3 , xi_4,  xi_5,  xi_6,  xi_7,  xi_8,  xi_9,  xi_10,  xi_11,  xi_12,  xi_13,   xi_f,
     x1_i,  x1_2,  x1_3 , x1_4,  x1_5,  x1_6,  x1_7,  x1_8,  x1_9,  x1_10,  x1_11,  x1_12,  x1_13,   x1_f,
     x2_i,  x2_1,  x2_3 , x2_4,  x2_5,  x2_6,  x2_7,  x2_8,  x2_9,  x2_10,  x2_11,  x2_12,  x2_13,   x2_f,
     x3_i,  x3_1,  x3_2 , x3_4,  x3_5,  x3_6,  x3_7,  x3_8,  x3_9,  x3_10,  x3_11,  x3_12,  x3_13,   x3_f,
     x4_i,  x4_1,  x4_2 , x4_3,  x4_5,  x4_6,  x4_7,  x4_8,  x4_9,  x4_10,  x4_11,  x4_12,  x4_13,   x4_f,
     x5_i,  x5_1,  x5_2 , x5_3,  x5_4,  x5_6,  x5_7,  x5_8,  x5_9,  x5_10,  x5_11,  x5_12,  x5_13,   x5_f,
     x6_i,  x6_1,  x6_2 , x6_3,  x6_4,  x6_5,  x6_7,  x6_8,  x6_9,  x6_10,  x6_11,  x6_12,  x6_13,   x6_f,
     x7_i,  x7_1,  x7_2 , x7_3,  x7_4,  x7_5,  x7_6,  x7_8,  x7_9,  x7_10,  x7_11,  x7_12,  x7_13,   x7_f,
     x8_i,  x8_1,  x8_2 , x8_3,  x8_4,  x8_5,  x8_6,  x8_7,  x8_9,  x8_10,  x8_11,  x8_12,  x8_13,   x8_f,
     x9_i,  x9_1,  x9_2 , x9_3,  x9_4,  x9_5,  x9_6,  x9_7,  x9_8,  x9_10,  x9_11,  x9_12,  x9_13,   x9_f,
    x10_i, x10_1, x10_2, x10_3, x10_4, x10_5, x10_6, x10_7, x10_8, x10_9,  x10_11, x10_12, x10_13,  x10_f,
    x11_i, x11_1, x11_2, x11_3, x11_4, x11_5, x11_6, x11_7, x11_8, x11_9,  x11_10, x11_12, x11_13,  x11_f,
    x12_i, x12_1, x12_2, x12_3, x12_4, x12_5, x12_6, x12_7, x12_8, x12_9,  x12_10, x12_11, x12_13,  x12_f,
    x13_i, x13_1, x13_2, x13_3, x13_4, x13_5, x13_6, x13_7, x13_8, x13_9,  x13_10, x13_11, x13_12,  x13_f,
     xf_i,  xf_1,  xf_2,  xf_3,  xf_4,  xf_5,  xf_6,  xf_7,  xf_8,  xf_9,   xf_10,  xf_11,  xf_12,  xf_13,
     xi_4_A, x2_3_A, x2_5_A, x3_2_A, x3_5_A, x4_i_A, x5_2_A, x5_3_A, x8_10_A, x8_11_A, x9_f_A, x10_8_A   ,
     x10_11_A, x11_8_A, x11_10_A, xf_9_A;
```

Figura 8: Definição das variáveis de decisão

3 Interpretação da solução ótima

Com todo o nosso modelo construído, falta apenas correr o programa em busca de uma solução. Assim, o resultado que obtemos através do LPSolve foi o seguinte:

Objective | Constraints | Sensitivity

Variables	MILP ...	MILP ...	MILP ...	MILP ...	MILP ...	re...
	108,03	105,23	104,62	102,02	101,41	101,41
x13_6	1	1	1	2	2	2
xi_1	1	1	1	1	1	1
xi_4	1	1	1	1	1	1
x1_2	1	1	1	1	1	1
x2_3	1	1	1	1	1	1
x2_5	1	1	1	1	1	1
x3_4	1	1	1	1	1	1
x3_5	1	1	1	1	1	1
x4_7	1	1	1	1	1	1
x5_13	1	1	1	1	1	1
x6_9	1	1	1	1	1	1
x7_12	1	1	1	1	1	1
x7_13	1	1	1	1	1	1
x8_10	1	1	1	1	1	1
x9_10	1	1	1	1	1	1
x9_f	1	1	1	1	1	1
x10_11	1	1	1	1	1	1
x12_f	1	1	1	1	1	1
x1_i	0	0	0	1	1	1
x6_1	1	1	1	1	1	1
x4_3	1	1	1	1	1	1
x11_7	0	0	1	0	1	1
x11_8	0	1	1	1	1	1
x10_9	1	1	1	1	1	1
x12_11	0	0	1	0	1	1
xf_12	0	0	1	0	1	1
x5_2_A	1	1	1	1	1	1
xi_2	0	0	0	0	0	0
xi_3	0	0	0	0	0	0

Figura 9: Output devolvido no `LPSolve`

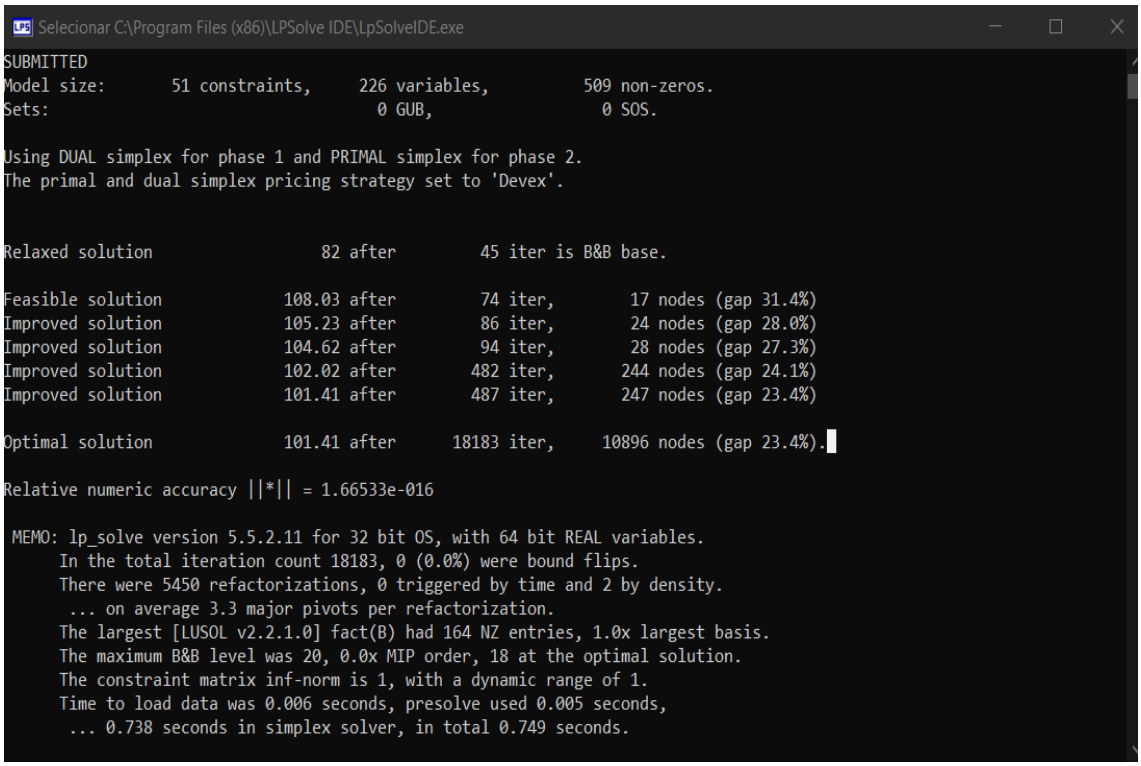


Figura 10: Output devolvido no terminal do `LPSolve`

Analisando, verificamos que a solução ótima encontrada foi um percurso com custo de 101.41 unidades de comprimento e que esta foi descoberta em 487 iterações e confirmada após 18183. A melhor solução, embora impossível, seria a passagem por todas as arestas obrigatórias sem repetir nenhuma e sem utilizar nenhum caminho aéreo, teria de custo 82.

Recriando o grafo com as arestas utilizadas na solução ótima temos:

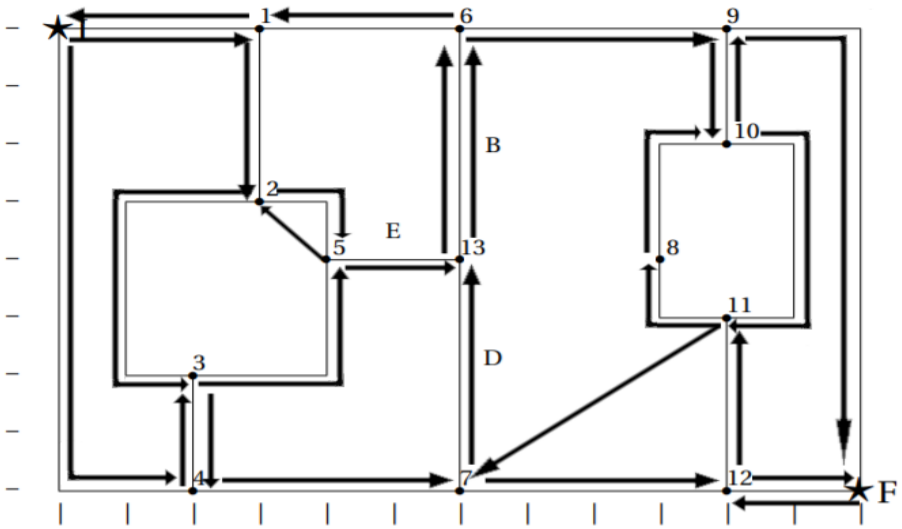


Figura 11: Grafo com todas as arestas da solução

4 Processos de validação do modelo

De modo a testar a validade da solução, testámos construir um caminho utilizando todas as arestas do resultado.

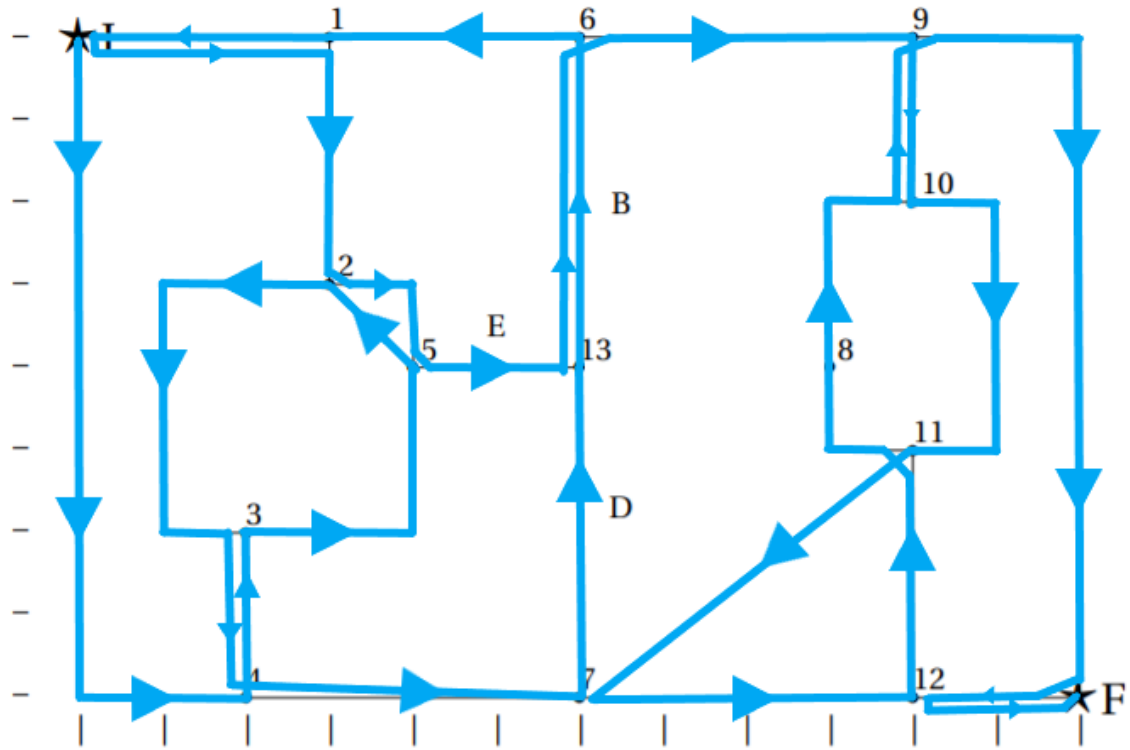


Figura 12: Possível caminho solução

Este é apenas um dos possíveis caminhos-solução. No entanto, através desta visualização, podemos mais facilmente notar onde houve algum excesso no caminho - através da dupla passagem em 5 arestas somadas com o uso de duas arestas aéreas.

5 Conclusão

O trabalho desenvolvido tinha como objetivo encontrar o percurso que obrigasse o *drone* a percorrer todas as linhas de alta tensão, minimizando o custo.

Inicialmente, analisamos o problema de forma a identificar algumas propriedades que facilitassem o processo de modelação. Depois, passamos para a definição do modelo matemático, onde identificamos o nosso grafo e as diversas restrições através de um modelo de Programação Linear. Por fim, passamos o modelo construído para o `LPSolve`, que nos devolveu a solução ótima para o modelo construído.

Acreditamos que será possível resolver outros problemas do mesmo tipo (**Problema do Carteiro Chinês**) com este modelo matemático elaborado, precisando apenas de alterar as restrições e a estrutura do grafo.

Assim, com a realização deste primeiro trabalho prático, conseguimos aprofundar os conhecimentos obtidos nas aulas teóricas e práticas de Programação Linear. Cremos que ganhamos experiência na utilização de um *software* nunca antes utilizado.