Universidade do Minho

Métodos Determinísticos de Investigação Operacional

MIEI - 3º ano - 1º semestre

Universidade do Minho

Trabalho Prático Experimental I

André Ferreira
A64296

Tiago Gomes
A69853

Sebastião Freitas
A71074

Gonçalo Almeida
A84610

Angélica Cunha
A84398

Braga, 13 de novembro de 2020

Conteúdo

Lista de Figuras

1. Introdução

Para este primeiro projeto, desenvolvido no âmbito da unidade curricular de Modelos Determinísticos de Investigação Operacional, é apresentado um problema comum de minimização, em que se pretende encontrar um percurso ótimo que percorra todo um conjunto de pontos ou arestas, minimizando a distância total percorrida.

Mais especificamente, temos o caso de um veículo não tripulado com necessidade de verificar linhas de energia eléctrica em alta tensão para verificar se há vegetação a interferir com as linhas. Sendo este veículo um drone, é possível fazer reposicionamentos entre vértices sem ter de seguir arestas predeterminadas, correspondentes a um deslocamento aéreo entre pontos seguindo o trajeto mais curto, mas mantém-se a necessidade de verificar todas as linhas, pelo menos uma vez.

De modo a apresentar uma solução a este problema, procedemos a modelá-lo, de modo a poder descrevê-lo de um modo mais facilmente compreensível e objetivo, permitindo-nos seguidamente aplicar métodos de programação linear, com o auxílio de software, para encontrar uma solução ótima para o problema.

2. Modelação

De forma a respondermos às necessidades do enunciado proposto, iniciamos por realizar as alterações necessárias à rede de distribuição de electricidade, tendo em conta o número mecanográfico 84610, seguido da formulação do modelo para atingir o objectivo proposto de minimização da distância total percorrida, garantindo que todas as arestas serão percorridas.

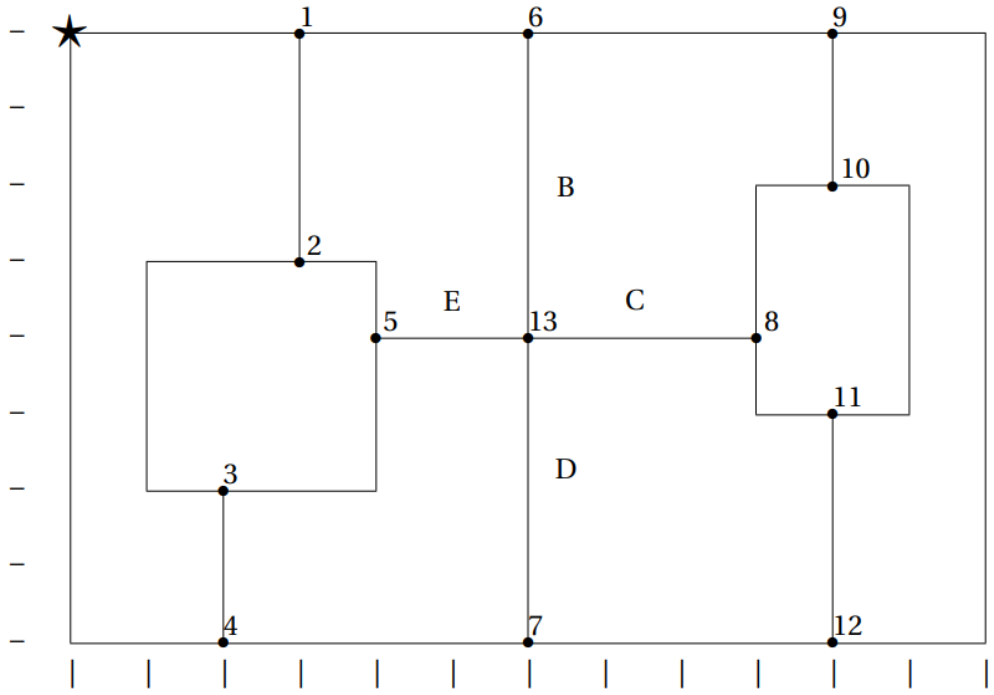

Figura 2.1: Rede original

2.1 Remoção de Arestas

Como proposto pelo enunciado, começamos por alterar a rede com as devidas arestas removidas. Teremos então que remover a aresta B, C e D (número 84610).

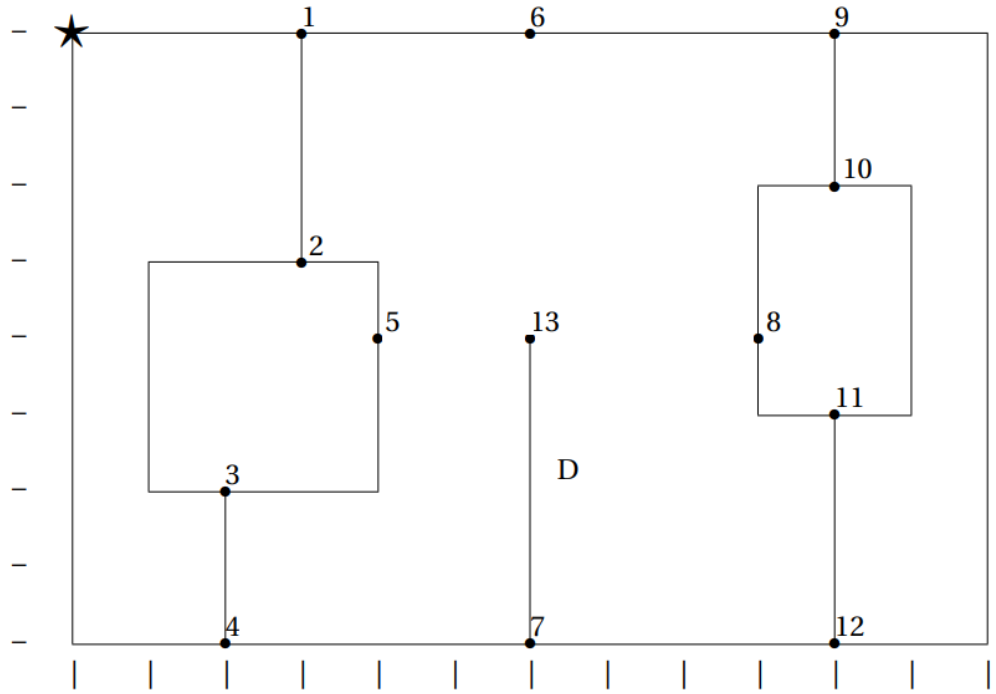

Figura 2.2: Nova rede

2.2 Formulação do Problema

O problema consiste na minimização do custo, a nível de distância, do percurso do drone de forma a percorrer todas as arestas necessárias pelo menos uma vez.

As distancias euclidianas estão definidas da seguinte forma:

			0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	8	0	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
3	8	1	3.00	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
3	5	2	4.24	3.00	0.00	3.16	5.10	1.41	4.24	5.83	6.08	7.62	7.07	7.28	8.60	3.16
2	2	3	6.32	6.08	3.16	0.00	2.00	2.83	7.21	4.47	7.28	10.00	8.94	8.06	8.25	4.47
2	0	4	8.25	8.06	5.10	2.00	0.00	4.47	8.94	4.00	8.06	11.31	10.00	8.54	8.00	5.66
4	4	5	5.66	4.12	1.41	2.83	4.47	0.00	4.47	4.47	5.00	7.21	6.32	6.08	7.21	2.00
6	8	6	6.00	3.00	4.24	7.21	8.94	4.47	0.00	8.00	5.00	4.00	4.47	6.40	8.94	4.00
6	0	7	10.00	8.54	5.83	4.47	4.00	4.47	8.00	0.00	5.00	8.94	7.21	5.00	4.00	4.00
9	4	8	9.85	7.21	6.08	7.28	8.06	5.00	5.00	5.00	0.00	4.12	2.24	1.41	4.12	3.00
10	8	9	10.00	7.00	7.62	10.00	11.31	7.21	4.00	8.94	4.12	0.00	2.00	5.00	8.00	5.66
10	6	10	10.20	7.28	7.07	8.94	10.00	6.32	4.47	7.21	2.24	2.00	0.00	3.00	6.00	4.47
10	3	11	11.18	8.60	7.28	8.06	8.54	6.08	6.40	5.00	1.41	5.00	3.00	0.00	3.00	4.12
10	0	12	12.81	10.63	8.60	8.25	8.00	7.21	8.94	4.00	4.12	8.00	6.00	3.00	0.00	5.66
6	4	13	7.21	5.00	3.16	4.47	5.66	2.00	4.00	4.00	3.00	5.66	4.47	4.12	5.66	0.00

Figura 2.3: Distâncias Euclidianas

Para alcançar o objectivo proposto, optamos por definir as seguintes variáveis de decisão:

$x_{ij} \in \mathbb{N}$: número de vezes que é percorrido o trajecto direto mais curto com origem no vértice i e destino j. Para vértices adjacentes, esta variável corresponde frequentemente à aresta entre os mesmos.

$x_{ij}a \in \mathbb{N}$: número de vezes que é percorrida a aresta com origem em i e destino j, para os casos em que a aresta entre i e j adjacentes não corresponde ao trajeto mais curto possível entre estes.

2.2.1 Função objectivo

Sendo o nosso objetivo encontrar uma solução ótima para o problema de percorrer todas as arestas, concluímos que se trata de um problema de minimização, onde estamos a tentar encontrar o caminho mais curto possível, ou seja, com a mínima distância total. Para tal, formulamos a seguinte função objetivo:

min: $\sum_{i=0}^{k} n_i * x_i$

Onde i corresponde a um trajeto específico entre dois vértices, n_i à distância correspondente a esse trajeto, e x_i o número de vezes que esse trajeto é percorrido, com k o número total de trajetos disponíveis.

2.2.2 Restrições

De modo a certificar que iremos obter uma solução correta, tivemos também de considerar as restrições a serem usadas no nosso modelo. Para tal, implementamos as seguintes restrições:

Arestas todas percorridas

$x_{ij} + x_{ji} >= 1$, para todo x_{ij} correspondente a uma aresta, entre i e j. Esta restrição serve para certificar que todas as arestas são percorridas, pelo menos uma vez, independentemente da direção;

Entrada e Saída de Vértices

$\sum_{i=0}^{k} x_{ij} = \sum_{i=0}^{k} x_{ji}$, $k \in \mathbb{N}$. Com esta restrição, pretendemos garantir que para cada caminho com destino numa aresta, existe um com origem dessa mesma aresta. Com isto garantimos um caminho aceitável, e que é possível de percorrer.

2.3 Modelação em LpSolve (input)

2.3.1 Função objetivo:

Transferimos a nossa função objetivo para lpsolve, com o objetivo de minimizar o custo total do percurso.

```
min: 3.00 x0001 + 4.24 x0002 + 6.32 x0003 + 8.25 x0004 + 5.66 x0005 + 6.00 x0006 + 10.00 x0007 + 9.85 x0008 + 10.00 x0009 + 10.20 x0010 + 11.18
x0011 + 12.81 x0012 + 7.21 x0013 + 3.00 x0102 + 6.08 x0103 + 8.06 x0104 + 4.12 x0105 + 3.00 x0106 + 8.54 x0107 + 7.21 x0108 + 7.00 x0109 + 7.28
x0110 + 8.60 x0111 + 10.63 x0112 + 5.00 x0113 + 3.16 x0203 + 5.10 x0204 + 1.41 x0205 + 4.24 x0206 + 5.83 x0207 + 6.08 x0208 +   7.62 x0209 + 7.
07 x0210 + 7.28 x0211 + 8.60 x0212 + 3.16 x0213 + 2.00 x0304 + 2.83 x0305 + 7.21 x0306 + 4.47 x0307 + 7.28 x0308 + 10.00 x0309 + 8.94 x0310 + 8.
06 x0311 + 8.25 x0312 + 4.47 x0313 + 4.47 x0405 + 8.94 x0406 + 4.00 x0407 + 8.06 x0408 + 11.31 x0409 + 10.00 x0410 + 8.54 x0411 + 8.00 x0412 +
5.66 x0413 + 4.47 x0506 + 4.47 x0507 + 5.00 x0508 + 7.21 x0509 + 6.32 x0510 + 6.08 x0511 + 7.21 x0512 + 2.00 x0513 + 8.00 x0607 + 5.00 x0608 +
4.00 x0609 + 4.47 x0610 +  6.40 x0611 + 8.94 x0612 + 4.00 x0613 + 5.00 x0708 + 8.94 x0709 + 7.21 x0710 + 5.00 x0711 + 4.00 x0712 + 4.00 x0713 +
4.12 x0809 + 2.24 x0810 + 1.41 x0811 + 4.12 x0812 + 3.00 x0813 + 2.00 x0910 + 5.00 x0911 + 8.00 x0912 + 5.66 x0913 + 3.00 x1011 + 6.00 x1012 +
4.47 x1013 + 3.00 x1112 + 4.12 x1113 + 5.66 x1213 + 3.00 x0100 + 4.24 x0200 + 6.32 x0300 + 8.25 x0400 + 5.66 x0500 + 6.00 x0600 + 10.00 x0700 +
9.85 x0800 + 10.00 x0900 + 10.20 x1000 + 11.18 x1100 + 12.81 x1200 + 7.21 x1300 + 3.00 x0201 + 6.08 x0301 + 8.06 x0401 + 4.12 x0501 + 3.00
x0601 + 8.54 x0701 + 7.21 x0801 + 7.00 x0901 + 7.28 x1001 + 8.60 x1101 + 10.63 x1201 + 5.00 x1301 + 3.16 x0302 + 5.10 x0402 + 1.41 x0502 + 4.24
x0602 + 5.83 x0702 + 6.08 x0802 + 7.62 x0902 + 7.07 x1002 + 7.28 x1102 + 8.60 x1202 + 3.16 x1302 + 2.00 x0403 + 2.83 x0503 + 7.21 x0603 + 4.47
x0703 + 7.28 x0803 + 10.00 x0903 + 8.94 x1003 + 8.06 x1103 + 8.25 x1203 + 4.47 x1303 + 4.47 x0504 + 8.94 x0604 + 4.00 x0704 + 8.06 x0804 + 11.
31 x0904 + 10.00 x1004 + 8.54 x1104 + 8.00 x1204 + 5.66 x1304 + 4.47 x0605 + 4.47 x0705 + 5.00 x0805 + 7.21 x0905 + 6.32 x1005 + 6.08 x0105 + 7.
21 x1205 + 2.00 x1305 + 8.00 x0706 + 5.00 x0806 + 4.00 x0906 + 4.47 x1006 + 6.40 x1106 + 8.94 x1206 + 4.00 x1306 + 5.00 x0807 + 8.94 x0907 + 7.
21 x1007 + 5.00 x1107 + 4.00 x1207 + 4.00 x0713 + 4.12 x0908 + 2.24 x1008 + 1.41 x1108 + 4.12 x1208 + 3.00 x1308 + 2.00 x1009 + 5.00 x1109 + 8.
00 x1209 + 5.66 x1309 + 3.00 x1110 + 6.00 x1210 + 4.47 x1310 + 3.00 x1211 + 4.12 x1311 + 5.66 x1312 + 2.00 x0205a + 2.00 x0502a + 6.00 x0203a +
6.00 x0302a + 4.00 x0503a + 4.00 x0305a + 3.00 x0810a + 3.00 x1008a + 5.00 x1011a + 5.00 x1110a + 2.00 x0811a + 2.00 x1108a + 10.00 x0400a + 10.
00 x0004a + 12.00 x0912a + 12.00 x1209a + 6.08 x1105 + 4.00 x1307;
```

Figura 2.4: Função objectivo

2.3.2 Restrições

Arestas todas percorridas

Foram adicionadas ao LpSolve as restrições que certificam que todas as arestas são percorridas, pelo menos uma vez, em qualquer direção.

```
x0001 + x0100 >= 1;
x0004a + x0400a >= 1;
x0102 + x0201 >= 1;
x0106 + x0601 >= 1;
x0906 + x0609 >= 1;
x0910 + x1009 >= 1;
x0912a + x1209a >= 1;
x0811a + x1108a >= 1;
x0810a + x1008a >= 1;
x1112 + x1211 >= 1;
x0713 + x1307 >= 1;
x0712 + x1207 >= 1;
x0304 + x0403 >= 1;
x0305a + x0503a >= 1;
x0205a + x0502a >= 1;
x0203a + x0302a >= 1;
x1011a + x1110a >= 1;
x0407 + x0704 >= 1;
```

Figura 2.5: Restrições para as arestas serem todas percorridas

Entrada e Saída de Vértices

Foram adicionadas também restrições para garantir que o número de entradas num vértice corresponde ao número de saídas do mesmo.

```
// Restrições para garantir a entrada e saída nos vértices, isto é, que existe um percurso lógico.

x0100 + x0200 + x0300 + x0400 + x0400a + x0500 + x0600 + x0700 + x0800 + x0900 + x1000 + x1100 + x1200 + x1300 = x0001 + x0002 + x0003 + x0004
+ x0004a + x0005 + x0006 + x0007 + x0008 + x0009 + x0010 + x0011 + x0012 + x0013;

x0001 + x0201 + x0301 + x0401 + x0501 + x0601 + x0701 + x0801 + x0901 + x1001 + x1101 + x1201 + x1301 = x0100 + x0102 + x0103 + x0104 + x0105 +
x0106 + x0107 + x0108 + x0109 + x0110 + x0111 + x0112 + x0113;

x0002 + x0102 + x0302 + x0302a + x0402 + x0502 + x0502a + x0602 + x0702 + x0802 + x0902 + x1002 + x1102 + x1202 + x1302 = x0200 + x0201 + x0203
+ x0203a + x0204 + x0205 + x0205a + x0206 + x0207 + x0208 + x0209 + x0210 + x0211 + x0212 + x0213;

x0003 + x0103 + x0203 + x0203a + x0403 + x0503 + x0503a + x0603 + x0703 + x0803 + x0903 + x1003 + x1103 + x1203 + x1303 = x0300 + x0301 + x0302
+ x0302a + x0304 + x0305 + x0305a + x0306 + x0307 + x0308 + x0309 + x0310 + x0311 + x0312 + x0313;

x0004 + x0004a + x0104 + x0204 + x0304 + x0504 + x0604 + x0704 + x0804 + x0904 + x1004 + x1104 + x1204 + x1304 = x0400 + x0400a + x0401 + x0402
+ x0403 + x0405 + x0406 + x0407 + x0408 + x0409 + x0410 + x0411 + x0412 + x0413;

x0005 + x0105 + x0205 + x0205a + x0305 + x0305a + x0405 + x0605 + x0705 + x0805 + x0905 + x1005 + x1105 + x1205 + x1305 = x0500 + x0501 + x0502
+ x0502a + x0503 + x0503a + x0504 + x0506 + x0507 + x0508 + x0509 + x0510 + x0511 + x0512 + x0513;

x0006 + x0106 + x0206 + x0306 + x0406 + x0506 + x0706 + x0806 + x0906 + x1006 + x1106 + x1206 + x1306 = x0600 + x0601 + x0602 + x0603 + x0604 +
x0605 + x0607 + x0608 + x0609 + x0610 + x0611 + x0612 + x0613;

x0007 + x0107 + x0207 + x0307 + x0407 + x0507 + x0607 + x0807 + x0907 + x1007 + x1107 + x1207 + x1307 = x0700 + x0701 + x0702 + x0703 + x0704 +
x0705 + x0706 + x0708 + x0709 + x0710 + x0711 + x0712 + x0713;

x0008 + x0108 + x0208 + x0308 + x0408 + x0508 + x0608 + x0708 + x0908 + x1008 + x1008a + x1108 + x1108a + x1208 + x1308 = x0800 + x0801 + x0802
+ x0803 + x0804 + x0805 + x0806 + x0807 + x0809 + x0810 + x0810a + x0811 + x0811a + x0812 + x0813;

x0009 + x0109 + x0209 + x0309 + x0409 + x0509 + x0609 + x0709 + x0809 + x1009 + x1109 + x1209 + x1209a + x1309 = x0900 + x0901 + x0902 + x0903
+ x0904 + x0905 + x0906 + x0907 + x0908 + x0910 + x0911 + x0912 + x0912a + x0913;

x0010 + x0110 + x0210 + x0310 + x0410 + x0510 + x0610 + x0710 + x0810 + x0810a + x0910 + x1110 + x1110a + x1210 + x1310 = x1000 + x1001 + x1002
+ x1003 + x1004 + x1005 + x1006 + x1007 + x1008 + x1008a + x1009 + x1011 + x1011a + x1012 + x1013;

x0011 + x0111 + x0211 + x0311 + x0411 + x0511 + x0611 + x0711 + x0811 + x0811a + x0911 + x1011 + x1011a + x1211 + x1311 = x1100 + x1101 + x1102
+ x1103 + x1104 + x1105 + x1106 + x1107 + x1108 + x1108a + x1109 + x1110 + x1110a + x1112 + x1113;

x0012 + x0112 + x0212 + x0312 + x0412 + x0512 + x0612 + x0712 + x0812 + x0912 + x0912a + x1012 + x1112 + x1312 = x1200 + x1201 + x1202 + x1203
+ x1204 + x1205 + x1206 + x1207 + x1208 + x1209 + x1209a + x1210 + x1211 + x1213;

x0013 + x0113 + x0213 + x0313 + x0413 + x0513 + x0613 + x0713 + x0813 + x0913 + x1013 + x1113 + x1213 = x1300 + x1301 + x1302 + x1303 + x1304 +
x1305 + x1306 + x1307 + x1308 + x1309 + x1310 + x1311 + x1312;
```

Figura 2.6: Restrição para entrada e saída

2.3.3 Declaração de variáveis

Declaração de variáveis no lpsolve como do tipo inteiro.

```
int x0001, x0002, x0003, x0004, x0005, x0006, x0007, x0008, x0009, x0010, x0011, x0012, x0013,
    x0100, x0102, x0103, x0104, x0105, x0106, x0107, x0108, x0109, x0110, x0111, x0112, x0113,
    x0200, x0201, x0203, x0204, x0205, x0206, x0207, x0208, x0209, x0210, x0211, x0212, x0213,
    x0300, x0301, x0302, x0304, x0305, x0306, x0307, x0308, x0309, x0310, x0311, x0312, x0313,
    x0400, x0401, x0402, x0403, x0405, x0406, x0407, x0408, x0409, x0410, x0411, x0412, x0413,
    x0500, x0501, x0502, x0503, x0504, x0506, x0507, x0508, x0509, x0510, x0511, x0512, x0513,
    x0600, x0601, x0602, x0603, x0604, x0605, x0607, x0608, x0609, x0610, x0611, x0612, x0613,
    x0700, x0701, x0702, x0703, x0704, x0705, x0706, x0708, x0709, x0710, x0711, x0712, x0713,
    x0800, x0801, x0802, x0803, x0804, x0805, x0806, x0807, x0809, x0810, x0811, x0812, x0813,
    x0900, x0901, x0902, x0903, x0904, x0905, x0906, x0907, x0908, x0910, x0911, x0912, x0913,
    x1000, x1001, x1002, x1003, x1004, x1005, x1006, x1007, x1008, x1009, x1011, x1012, x1013,
    x1100, x1101, x1102, x1103, x1104, x1105, x1106, x1107, x1108, x1109, x1110, x1112, x1113,
    x1200, x1201, x1202, x1203, x1204, x1205, x1206, x1207, x1208, x1209, x1210, x1211, x1213,
    x1300, x1301, x1302, x1303, x1304, x1305, x1306, x1307, x1308, x1309, x1310, x1311, x1312,
    x0203a, x0302a, x0205a, x0502a, x0305a, x0503a, x0810a, x1008a, x1011a, x1110a, x0811a, x1108a,
    x0004a, x0400a, x0912a, x1209a;
```

Figura 2.7: Declaração de variáveis

2.4 Outputs

Nas figuras seguintes encontram-se os diversos outputs obtidos pelo modelo.

2.4.1 Output da solução

Na figura abaixo encontra-se o output do modelo com possibilidades de percursos, tendo sido encontrado o percurso óptimo

Variables	MILP ...	MILP ...	MILP ...	MILP ...	MILP ...	MILP ...	MILP ...	MILP ...	MILP ...	re...
	99.81	99.16	97.63	97.37	97.16	96.47	95.16	94	91.12	91.12
x0910	1	1	1	2	1	1	1	1	2	2
x0001	1	1	1	1	1	1	1	1	1	1
x0102	1	1	1	1	1	1	1	1	1	1
x0106	1	1	1	1	1	1	1	1	1	1
x0304	2	2	1	1	1	1	1	1	1	1
x0609	1	1	1	1	1	1	0	1	1	1
x0712	1	1	1	1	0	0	1	0	1	1
x1112	1	1	1	1	1	1	1	1	1	1
x1113	0	0	0	1	0	0	0	0	1	1
x0201	0	0	0	1	0	1	0	1	1	1
x0403	0	0	0	0	0	0	0	1	1	1
x0704	0	0	1	1	1	1	1	1	1	1
x1207	1	1	0	0	1	1	0	1	1	1
x0205a	1	1	1	1	1	1	1	1	1	1
x0302a	0	0	0	1	0	1	0	1	1	1
x0503a	1	1	1	1	1	1	1	1	1	1
x1008a	0	0	0	1	0	0	1	0	1	1
x1011a	1	1	1	1	1	1	0	1	1	1
x0811a	1	0	0	1	0	0	1	0	1	1
x0400a	1	1	1	1	1	1	1	1	1	1
x1209a	1	1	1	1	0	0	1	0	1	1
x1307	0	0	1	1	0	0	1	1	1	1

Figura 2.8: Percurso ótimo

2.4.2 Output Lpsolve

```
Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.


Relaxed solution                78 after         32 iter is B&B base.

Feasible solution            99.81 after         69 iter,        15 nodes (gap 27.6%)
Improved solution            99.16 after         70 iter,        16 nodes (gap 26.8%)
Improved solution            97.63 after         86 iter,        23 nodes (gap 24.8%)
Improved solution            97.37 after        112 iter,        35 nodes (gap 24.5%)
Improved solution            97.16 after        262 iter,       120 nodes (gap 24.3%)
Improved solution            96.47 after        369 iter,       187 nodes (gap 23.4%)
Improved solution            95.16 after        810 iter,       454 nodes (gap 21.7%)
Improved solution               94 after       1080 iter,       617 nodes (gap 20.3%)
Improved solution            91.12 after       1133 iter,       646 nodes (gap 16.6%)

Optimal solution             91.12 after       3134 iter,      1906 nodes (gap 16.6%).

Relative numeric accuracy ||*|| = 1.66533e-016

 MEMO: lp_solve version 5.5.2.5 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 3134, 0 (0.0%) were bound flips.
      There were 955 refactorizations, 0 triggered by time and 2 by density.
       ... on average 3.3 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 128 NZ entries, 1.0x largest basis.
      The maximum B&B level was 18, 0.0x MIP order, 11 at the optimal solution.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0.024 seconds, presolve used 0.015 seconds,
       ... 0.196 seconds in simplex solver, in total 0.235 seconds.
```

Figura 2.9: Output produzido

2.4.3 Exemplo de Percurso

Como foi possível verificar anteriormente, o drone percorre todas as arestas necessárias, efetuando também um trajeto entre 11 e 13, onde não existe aresta. A distância total percorrida para este percurso, como indicada pelo lpsolve, é de 91.12.

Na figura seguinte encontra-se um exemplo de um percurso possível (as setas mudam de cor para melhorar a inspecção, de forma a tentar remover ambiguidade e mostrar mais claramente o percurso).

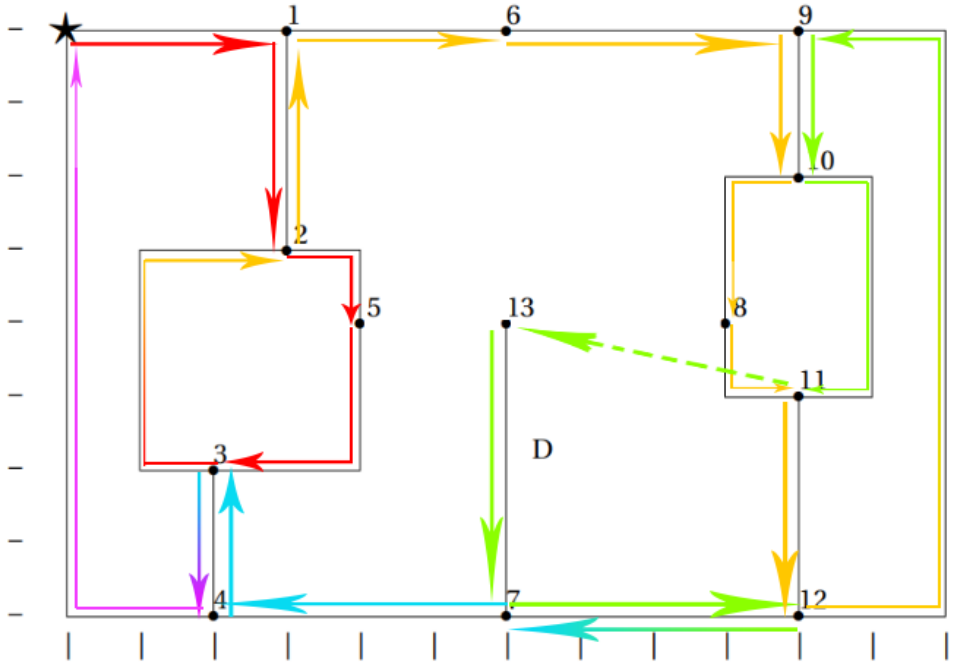

Figura 2.10: Exemplo circuito realizado

Desta forma validamos a solução óptima, mostrando que a solução proposta é admissível e correta, e vai ao encontro das necessidades propostas pelo enunciado, percorrendo todos os vértices e sendo um caminho válido.

3. Conclusão

Para a resolução deste problema, a modulação ocorreu de um modo bastante natural, partindo do mesmo como um problema de minimização, e chegamos rapidamente a uma função objetivo que satisfazia os nossos critérios, usando a tabela de distâncias entre as arestas.

Após conseguir construir todas as restrições necessárias para obter soluções aceitáveis, obtivemos uma modelação completa, e apenas necessitamos de aplicar a mesma ao *lpsolve* para obter uma solução ótima para o problema.

Este trabalho ajudou-nos a aprofundar o nosso uso de *solvers* para estes problemas, e mostrou-nos também a importância da modelação acima de tudo, e de conseguir transformar um problema em funções e restrições que o descrevem concretamente e sem perder nenhuma informação, de modo a conseguir obter soluções ótimas para problemas mais complexos.