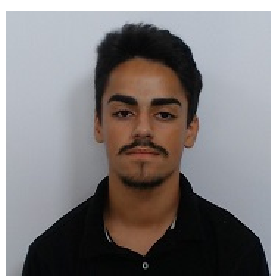
Universidade do Minho
Escola de Engenharia

Trabalho 1

Métodos Determinísticos de Investigação Operacional

Bruno Filipe de Sousa Dias A89583
Guilherme Santiago Lopes Pereira A89479
Luís Enes Sousa A89597
Pedro Miguel de Soveral Pacheco Barbosa A89529

13 de novembro de 2020

Conteúdo

1 Introdução

No âmbito da cadeira de Métodos Determinísticos de Investigação Operacional foi-nos proposta a resolução de um problema que consiste em determinar o percurso de um grafo em que todas as suas arestas são percorridas, pelo menos, uma vez, minimizando a distância total percorrida. As arestas do grafo podem ser percorridas em qualquer sentido, e pode ser necessário atravessar a mesma aresta mais do que uma vez.

Assim sendo, o problema a resolucionar é o do drone que ocorre quando um veículo não tripulado tem de inspeccionar linhas de transporte de energia eléctrica em alta tensão para verificar se há vegetação a interferir com as linhas. Essa distância percorrida pelo drone é conhecida como distância euclidiana.

Este trabalho prático 1 tem como principal objetivo avaliar a análise de sistemas e a criação de modelos que os descrevam e resolvam utilizando, para isso, programas adequados para conseguirmos realizar as tarefas propostas.

2 Resolução dos Exercícios Propostos - Parte I

2.1 Exercício 0

2.1.1 Remoção de arestas

As indicações dadas para a remoção de arestas foram as seguintes. Haja cinco arestas ABCDE estas deverão ser removidas da seguinte forma:

- se B par, remover a aresta B;
- se C par, remover a aresta C;
- se D par, remover a aresta D;
- se E par, remover a aresta E;

Sendo que o valor de ABCDE equivale ao número de inscrição do estudante do grupo com maior número de inscrição, o valor do nosso ABCDE é de 89597. Deste modo, e analisando as diferentes indicações dadas, chegamos à conclusão que não deveríamos retirar nenhuma das arestas do grafo.

Assim sendo, a rede não terá nenhuma das suas arestas removida e apresenta-se de igual forma àquela dada no enunciado, que é a seguinte:

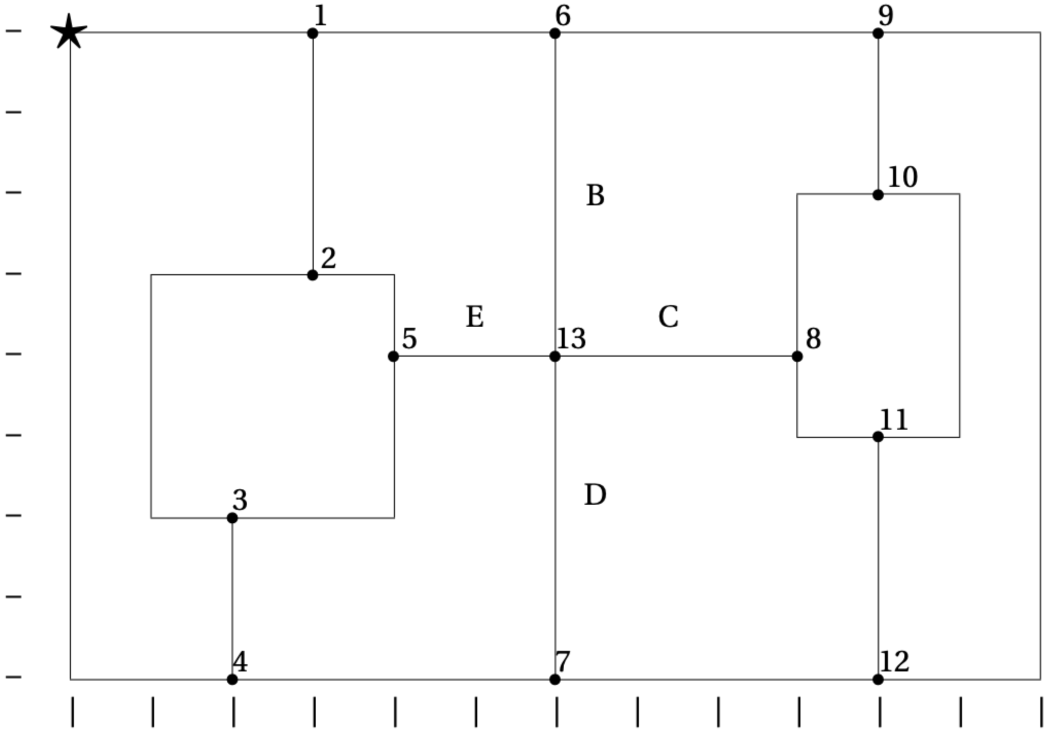


Figura 1: Mapa de Linhas

2.2 Exercício 1

2.2.1 Descrição do Problema

O problema que pretendemos solucionar incide quando um veículo não tripulado (Drone) tem de inspeccionar linhas de transporte de energia eléctrica em alta tensão para verificar se há vegetação a interferir com as linhas. Para que tal aconteça, o drone terá de verificar todas as linhas para verificar que tudo se encontra sem vegetação. Assim, entramos num problema em tudo idêntico ao problema do carteiro chinês, uma vez que o drone necessita de precorrer todas as linhas do mapa sem exceção, pelo menos uma vez e voltar no final, ao seu ponto de partida (analogamente ao carteiro que tinha de percorrer todas as ruas da cidade).

Para que tal aconteça, é necessário ter como premissa, que todos os vértices do mapa tenham um número par de linhas incidentes. As linhas do mapa entre vértices terão também uma distância associada, algo indispensável, e que será fulcral para o melhor percurso ser alcançado.

2.2.2 Objetivo do Problema

O objetivo do problema será alcançar a melhor rota e o melhor conjunto de retas que ligam os vários vértices, de modo ao drone poder averiguar o estado de todas as linhas de transporte de energia eléctrica em alta tensão, e ver se as mesmas se encontram com qualquer tipo de obstrução ou molestamento por parte de possível vegetação. A ideia é que o drone consiga fazer a averiguação de todas as linhas na menor distância euclidiana total possível.

2.2.3 Variáveis de Decisão

Para iniciar a formulação do problema de programação linear apresentado é necessário definir as variáveis de decisão. Neste caso específico, as variáveis de decisão irão representar as ligações adicionais existentes entre os vértices ímpares, com o objetivo de criar um grafo em que todos os seus vértices têm grau par. Consequentemente, as variáveis de decisão vão ser **variáveis binárias** que indicam se a aresta entre dois vértices de grau ímpar é ou não selecionada e adicionada ao precurso. Assim, estas variáveis irão ter o valor de 1 se forem selecionadas para o percurso, e terão o valor de 0 caso contrário.

Deste modo as nossas variáveis de decisão irão ser:

Xij: Aresta adicional entre o vértice i e o vértice j (binário).

i,j $\in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$

Neste conjunto, a que **i** e **j** pertencem, estão então todos os números que representam vértices com um número ímpar de linhas incidentes, uma vez que o objetivo é tornar todos os vértices de grau ímpar em vértices de grau par de modo a conseguirmos solucionar o problema.

Como o objetivo é tornar o número de incidência dos vértices ímpares em pares, de forma a construir um percurso para o drone, pretendemos adicionar uma aresta aos vértices ímpares, não alterando o vértice par, que neste caso é o 13.

2.2.4 Dados

Os dados apresentados para este problema constam no enunciado e passam pela representação da distância das linhas de alta tensão. As distâncias estão retratadas como comprimentos e distâncias eucladianas de arestas entre dois vértices do mapa.

Avaliando o grafo após a remoção das arestas pedidas (que no nosso caso não foi removida nenhuma), constatamos que o comprimento e distância total de todas as arestas é de 85 centímetros. Nesta distância não constam, no entanto, as arestas que irão ser adicionadas para a resolução do problema.

2.2.5 Restrições

As restrições neste modelo vão determinar as diretrizes para a seleção das ligações e arestas adicionais. Iremos ter que cada vértice em estudo (vértices de grau ímpar) irá ter apenas um caminho adicional, e assim, a soma de todas as possibilidades de caminhos incidentes em si, irá ser igual a 1. Isto deve-se ao facto de estarmos a somar todas as variáveis binárias correspondentes a esses caminhos, e apenas 1 delas irá ser igual a 1, e todas as restantes irão ser iguais a 0, uma vez que os caminhos que representam não serão adicionados ao mapa.

Além disso, todas as variáveis são declaradas como binárias, uma vez que pretendemos que estas sejam ou 1, caso sejam adicionadas, ou 0, caso contrário.

2.3 Exercício 2

2.3.1 Variáveis de Decisão

Como referido anteriormente, teremos então variáveis de decisão que correspondem a caminhos entre um par de vértices. Estas variáveis são então representadas como **Xij**, sendo **i** e **j** identificadores numerais dos vértices. É importante realçar que, como estamos a utilizar um sistema onde não são verificados sentidos, não iremos ter duas possibilidades de caminho, apenas uma. Por exemplo um caminho entre o vértice 2 e o vértice 6, tanto X26 como X62 representam um caminho entre entre os mesmos dois vértices, no entanto, como não é referido nenhum sistema de sentidos, isto torna-se redundante. Neste caso escolheríamos o caminho X26. Lembrando ainda que não estudamos casos em que existe um caminho que sai de um vértice e volte a si mesmo. Assim, nestes casos, referimos unicamente um caminho, sendo i sempre o vértice com identificador númerico mais baixo do que j.

Temos então que:

Xij: Aresta adicional entre o vértice i e o vértice j (binário).

Xij $\in \{0, 1\}$

i,j $\in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$

2.3.2 Restrições

Como referido acima, as restrições irão determinar a escolha de caminhos opcionais. Assim, teremos as seguintes restrições por vértices de grau ímpar:

Vértice 1:

x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x110 + x111 + x112 = 1;

Vértice 2:

x12 + x23 + x24 + x25 + x26 + x27 + x28 + x29 + x210 + x211 + x212 = 1;

Vértice 3:

x13 + x23 + x34 + x35 + x36 + x37 + x38 + x39 + x310 + x311 + x312 = 1;

Vértice 4:

x14 + x24 + x34 + x45 + x46 + x47 + x48 + x49 + x410 + x411 + x412 = 1;

Vértice 5:

x15 + x25 + x35 + x45 + x56 + x57 + x58 + x59 + x510 + x511 + x512 = 1;

Vértice 6:

x16 + x26 + x36 + x46 + x56 + x67 + x68 + x69 + x610 + x611 + x612 = 1;

Vértice 7:

x17 + x27 + x37 + x47 + x57 + x67 + x78 + x79 + x710 + x711 + x712 = 1;

Vértice 8:

x18 + x28 + x38 + x48 + x58 + x68 + x78 + x89 + x810 + x811 + x812 = 1;

Vértice 9:

x19 + x29 + x39 + x49 + x59 + x69 + x79 + x89 + x910 + x911 + x912 = 1;

Vértice 10:

x110 + x210 + x310 + x410 + x510 + x610 + x710 + x810 + x910 + x1011 + x1012 = 1;

Vértice 11:

x111 + x211 + x311 + x411 + x511 + x611 + x711 + x811 + x911 + x1011 + x1112 = 1;

Vértice 12:

x112 + x212 + x312 + x412 + x512 + x612 + x712 + x812 + x912 + x1012 + x1112 = 1;

Além destas restrições, temos ainda as restrições das variáveis binárias. Reparemos agora também, uma vez que é mais fácil a perceção visivelmente, no que foi referido acima no caso das variáveis que representam também o mesmo caminho, e que apenas existe uma delas. Teremos então para a declaração de variáveis:

bin
x12 x13 x14 x15 x16 x17 x18 x19 x110 x111 x112
x23 x24 x25 x26 x27 x28 x29 x210 x211 x212
x34 x35 x36 x37 x38 x39 x310 x311 x312
x45 x46 x47 x48 x49 x410 x411 x412
x56 x57 x58 x59 x510 x511 x512
x67 x68 x69 x610 x611 x612
x78 x79 x710 x711 x712
x89 x810 x811 x812
x910 x911 x912
x1011 x1012
x1112;

2.3.3 Função Objetivo

Com a função objetivo pretendemos alcançar a menor distância eucladiana que o drone precisa de percorrer para conseguir passar por todas as linhas de alta tensão do mapa. As arestas/linhas adicionais e as distâncias que estas vão adicionar ao mapa, para que o drone consiga percorrer todas as linhas do mapa, é a parte do problema e da sua resolução, que pretendemos resolver e aprimorar. Assim, a função objetivo passa por adicionar todas as distâncias das arestas do mapa inicial (que é equivalente a 85 centímetros) com a soma de todas as distâncias das arestas adicionais multiplicadas pela sua variável binária (que determina se foram escolhidas ou não). Assim, a função objetivo será:

min: 85 + 3.00 x12 + 6.08 x13 + 8.06 x14 + 4.12 x15 + 3.00 x16 + 8.54 x17 + 7.21 x18 + 7.00 x19 + 7.28 x110 + 8.60 x111 + 10.63 x112 + 3.16 x23 + 5.10 x24 + 1.41 x25 + 4.24 x26 + 5.83 x27 + 6.08 x28 + 7.62 x29 + 7.07 x210 + 7.28 x211 + 8.60 x212 + 2.00 x34 + 2.83 x35 + 7.21 x36 + 4.47 x37 + 7.28 x38 + 10.00 x39 + 8.94 x310 + 8.06 x311 + 8.25 x312 + 4.47 x45 + 8.94 x46 + 4.00 x47 + 8.06 x48 + 11.31 x49 + 10.0 x410 + 8.54 x411 + 8.00 x412 + 4.47 x56 + 4.47 x57 + 5.00 x58 + 7.21 x59 + 6.32 x510 + 6.08 x511 + 7.21 x512 + 8.00 x67 + 5.00 x68 + 4.00 x69 + 4.47 x610 + 6.40 x611 + 8.94 x612 + 5.00 x78 + 8.94 x79 + 7.21 x710 + 5.00 x711 +

4.00 x712 + 4.12 x89 + 2.24 x810 + 1.41 x811 + 4.12 x812 + 2.00 x910 + 5.00 x911 + 8.00 x912 + 3.00 x1011 + 6.00 x1012 + 3.00 x1112;

2.4 Exercício 3

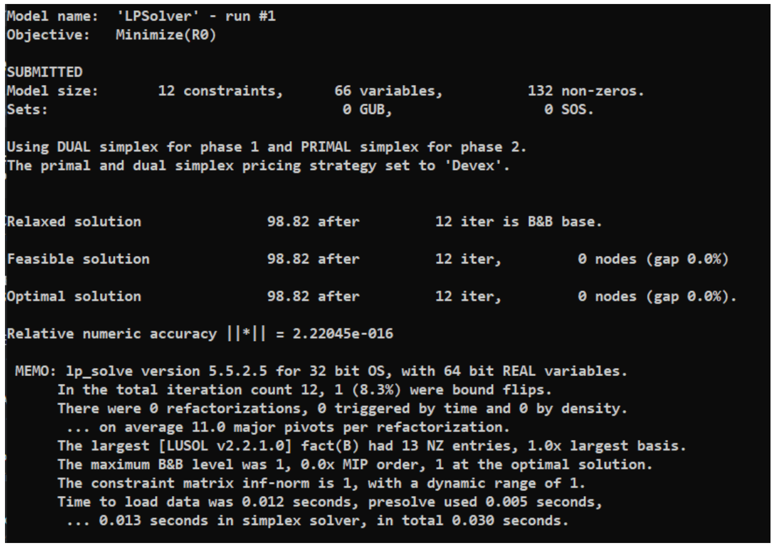
Utilizando o software LPSolve, e introduzindo as restrições, bem como a função objetivo, chegámos ao seguinte ficheiro:

```
/* Objective function */
min: 85 + 3.00 x12 + 6.08 x13 + 8.06 x14 + 4.12 x15 + 3.00 x16 + 8.54 x17 + 7.21 x18 + 7.00  x19 + 7.28 x110 + 8.60 x111 + 10.63 x112
                   + 3.16 x23 + 5.10 x24 + 1.41 x25 + 4.24 x26 + 5.83 x27 + 6.08 x28 + 7.62  x29 + 7.07 x210 + 7.28 x211 + 8.60  x212
                              + 2.00 x34 + 2.83 x35 + 7.21 x36 + 4.47 x37 + 7.28 x38 + 10.00 x39 + 8.94 x310 + 8.06 x311 + 8.25  x312
                                         + 4.47 x45 + 8.94 x46 + 4.00 x47 + 8.06 x48 + 11.31 x49 + 10.0 x410 + 8.54 x411 + 8.00  x412
                                                    + 4.47 x56 + 4.47 x57 + 5.00 x58 + 7.21  x59 + 6.32 x510 + 6.08 x511 + 7.21  x512
                                                               + 8.00 x67 + 5.00 x68 + 4.00  x69 + 4.47 x610 + 6.40 x611 + 8.94  x612
                                                                          + 5.00 x78 + 8.94  x79 + 7.21 x710 + 5.00 x711 + 4.00  x712
                                                                                     + 4.12  x89 + 2.24 x810 + 1.41 x811 + 4.12  x812
                                                                                                 + 2.00 x910 + 5.00 x911 + 8.00  x912
                                                                                                             + 3.00 x1011 + 6.00 x1012
                                                                                                                          + 3.00 x1112;

/* Variable bounds */
x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x110 + x111 + x112 = 1;
x12 + x23 + x24 + x25 + x26 + x27 + x28 + x29 + x210 + x211 + x212 = 1;
x13 + x23 + x34 + x35 + x36 + x37 + x38 + x39 + x310 + x311 + x312 = 1;
x14 + x24 + x34 + x45 + x46 + x47 + x48 + x49 + x410 + x411 + x412 = 1;
x15 + x25 + x35 + x45 + x56 + x57 + x58 + x59 + x510 + x511 + x512 = 1;
x16 + x26 + x36 + x46 + x56 + x67 + x68 + x69 + x610 + x611 + x612 = 1;
x17 + x27 + x37 + x47 + x57 + x67 + x78 + x79 + x710 + x711 + x712 = 1;
x18 + x28 + x38 + x48 + x58 + x68 + x78 + x89 + x810 + x811 + x812 = 1;
x19 + x29 + x39 + x49 + x59 + x69 + x79 + x89 + x910 + x911 + x912 = 1;
x110 + x210 + x310 + x410 + x510 + x610 + x710 + x810 + x910 + x1011 + x1012 = 1;
x111 + x211 + x311 + x411 + x511 + x611 + x711 + x811 + x911 + x1011 + x1112 = 1;
x112 + x212 + x312 + x412 + x512 + x612 + x712 + x812 + x912 + x1012 + x1112 = 1;


bin
x12 x13 x14 x15 x16 x17 x18 x19 x110 x111 x112
    x23 x24 x25 x26 x27 x28 x29 x210 x211 x212
        x34 x35 x36 x37 x38 x39 x310 x311 x312
            x45 x46 x47 x48 x49 x410 x411 x412
                x56 x57 x58 x59 x510 x511 x512
                    x67 x68 x69 x610 x611 x612
                        x78 x79 x710 x711 x712
                            x89 x810 x811 x812
                                x910 x911 x912
                                     x1011 x1012
                                           x1112;
```

Figura 2: Input do Ficheiro

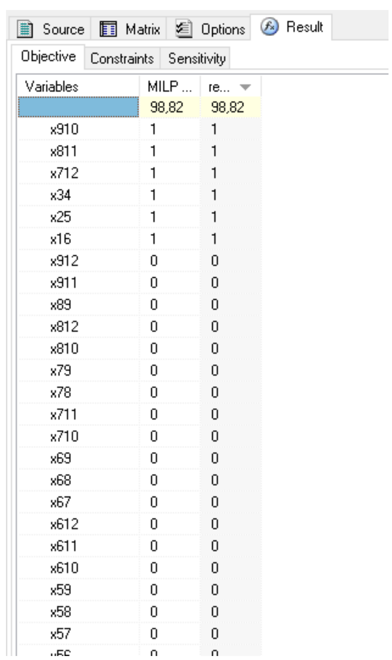
2.5 Exercício 4

Após o software LPSolve executar o input dado, foi gerada a solução ótima e foram selecionadas as melhores arestas adicionais para conseguirmos ter todos os vértices com grau par, com a menor diatância eucladiana possível. O output gerado pelo software foi o seguinte:



```
Model name:  'LPSolver' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size:       12 constraints,      66 variables,          132 non-zeros.
Sets:                                   0 GUB,                  0 SOS.

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


Relaxed solution               98.82 after        12 iter is B&B base.

Feasible solution              98.82 after        12 iter,         0 nodes (gap 0.0%)

Optimal solution               98.82 after        12 iter,         0 nodes (gap 0.0%).

Relative numeric accuracy ||*|| = 2.22045e-016

 MEMO: lp_solve version 5.5.2.5 for 32 bit OS, with 64 bit REAL variables.
      In the total iteration count 12, 1 (8.3%) were bound flips.
      There were 0 refactorizations, 0 triggered by time and 0 by density.
       ... on average 11.0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 13 NZ entries, 1.0x largest basis.
      The maximum B&B level was 1, 0.0x MIP order, 1 at the optimal solution.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0.012 seconds, presolve used 0.005 seconds,
       ... 0.013 seconds in simplex solver, in total 0.030 seconds.
```

Figura 3: Ficheiro de Output no Terminal



Variables	MILP ...	re...
	98,82	98,82
x910	1	1
x811	1	1
x712	1	1
x34	1	1
x25	1	1
x16	1	1
x912	0	0
x911	0	0
x89	0	0
x812	0	0
x810	0	0
x79	0	0
x78	0	0
x711	0	0
x710	0	0
x69	0	0
x68	0	0
x67	0	0
x612	0	0
x611	0	0
x610	0	0
x59	0	0
x58	0	0
x57	0	0

Figura 4: Ficheiro de Output no IDE

2.6 Exercício 5

Resolvido o problema e executado o input pelo software LPSolve, chegamos então a uma solução ótima. Pela análise dos outputs e da resposta, podemos ver o resultado obtido, os caminhos escolhidos (aqueles que a variável da aresta possui valor 1) e ainda o total de distância euclidiana a percorrer.

Assim, as arestas, ou, as ligações entre os vértices de grau ímpar escolhidas e obtidas na solução otíma são:

- **X16 -** Vértice 1 e Vértice 6 (3.00 de Distancia Eucladiana);
- **X25 -** Vértice 2 e Vértice 5 (1.41 de Distancia Eucladiana);
- **X34 -** Vértice 3 e Vértice 4 (2.00 de Distancia Eucladiana);
- **X712 -** Vértice 7 e Vértice 12 (4.00 de Distancia Eucladiana);
- **X811 -** Vértice 8 e Vértice 11 (1.41 de Distancia Eucladiana);
- **X910 -** Vértice 9 e Vértice 10 (2.00 de Distancia Eucladiana);

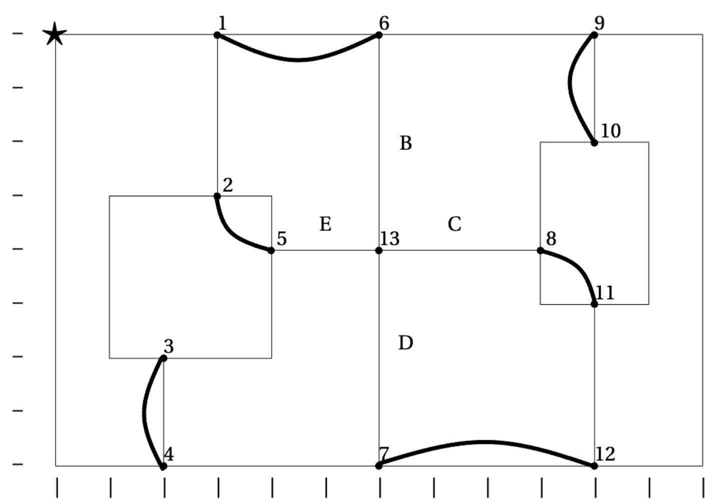


Figura 5: Percurso final do Drone (com arestas adicionais)

Acima representado está o percurso final do drone, onde já estão introduzidas as arestas adicioanis, necessárias para a resolução do problema, que pertencem à solução ótima. Relembre-se ainda que, como referido várias vezes anteriormente, as arestas adicionais não vêm particularizadas com qualquer tipo de sentido.

A solução ótima obtida, consoante as arestas adicionadas irá fazer com que o drone percorra um percurso de 98,82. Este valor obtém-se somando a distância eucladiana inicial de todas as arestas, que era 85, com a distância euclidiana de todas as arestas adicionadas na solução ótima. Desse modo, a distância do precurso do Drone seria:

Distância = 85.00 + 3.00 + 1.41 + 2.00 + 4.00 + 1.41 + 2.00 = **98.82**

2.7 Exercício 6

Obtida uma resposta e uma solução ótima, é então necessário passar por uma validação do modelo. Umas vez enunciadas algumas regras no problema, estas terão de ser respeitadas! Para além dessas, teremos ainda de respeitar regras que estão relacionadas com os grafos Neste caso teremos de verificar as seguintes peculiaridades:

- **Os vértices são todos pares:** Pela análise do percurso obtido, podemos verificar que todos os vértices têm um número de arestas incidentes par. As restrições impostas e a maneira como o problema foi estruturado fazem com que todos os vértices fiquem de grau par, logo a regra de todos os vértices serem de grau par é respeitada;
- **Todas as arestas são percorridas:** Pela análise da solução ótima, a distância percorrida pelo drone é igual a 98.82, que é igual a 85 (distância das arestas totais iniciais) mais 13.82 (adição de todas as distâncias das arestas adicionadas). Logo, garantimos que o drone passa em todas as arestas.
- **Regressa onde começou:** Pela análise da solução ótima, o drone, como verificado anteriormente, percorre todas as arestas e, como isto acontece, o Drone tem sempre o cenário possível de voltar ao seu ponto de partida. Dado que percorre todas as linhas, consegue voltar ao sítio onde começou, assim garantindo que o drone pode acabar o percurso no seu ponto de partida.