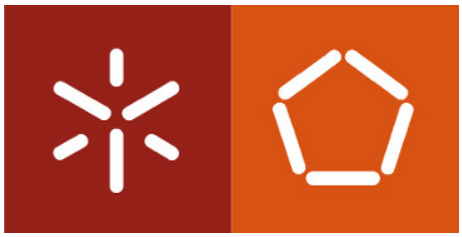
Modelos Determinísticos de Investigação Operacional

8 de Outubro de 2019

Grupo nr.	15
a83899	André Morais
a84577	José Pedro Silva
a85954	Luís Ribeiro
a84783	Pedro Rodrigues



Mestrado Integrado em Engenharia Informática
Universidade do Minho

Conteúdo

1 Introdução

O presente relatório irá abordar a elaboração do projeto realizado no âmbito da Unidade Curricular **Modelos determinísticos de Investigação Operacional**. Este consiste em determinar o circuito, em que todos os arcos de um grafo são percorridos pelo menos uma vez, tendo como objetivo a minimização da distância total percorrida.

Para tal, foi-nos fornecido um gráfico (referencia ao grafico) que inicialmente estava incompleto no que diz respeito à direção de alguns arcos.

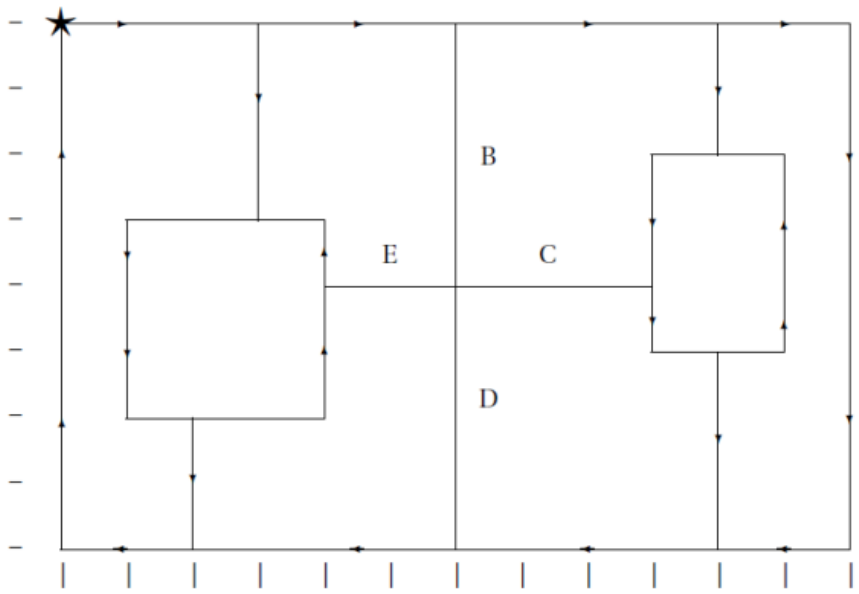


Figura 1: Gráfico Inicial

2 Questões

2.1 Questão 1

"Indique o valor de ABCDE e apresente a rede com indicação dos sentidos das ruas BDCE. Formule este problema como um modelo de programação linear: identifique claramente as variáveis de decisão e os dados, e o significado das restrições e da função objectivo. Teça comentários sobre o desenvolvimento do modelo (ver informação no Anexo)".

Sendo ABCDE o maior número mecanográfico do grupo, analisando os mesmos concluímos que *a85954* é o maior. Através do conjunto de regras definidas no enunciado do trabalho prático completamos o gráfico, obtendo o seguinte gráfico (Figura 2).

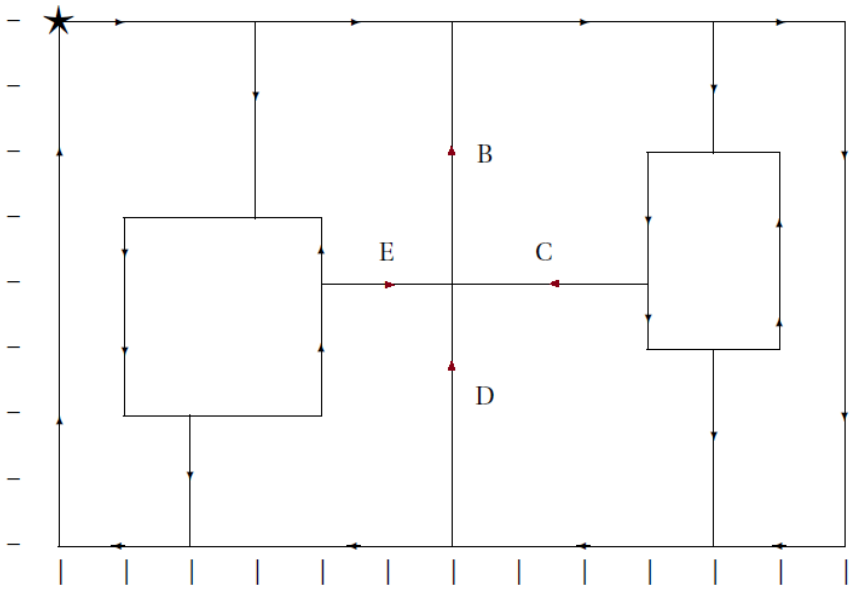


Figura 2: Gráfico com direções

2.1.1 Variáveis de decisão e dados

De modo a poder identificar os vértices para posteriormente poder etiquetar os arcos, o grupo atribui a seguinte representação ao gráfico (Figura 3).

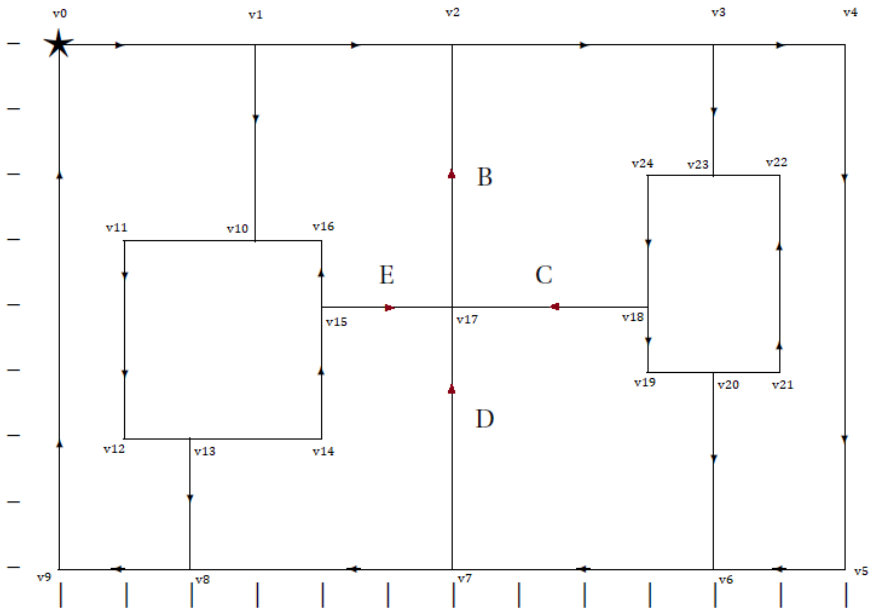


Figura 3: Gráfico com os vértices

1. **Variáveis de decisão**

   - Seja x_{ij} o número de vezes que um arco é percorrido, onde i corresponde ao vértice de origem e j ao vértice de chegada. Isto é, o arco x_{01} corresponde ao número que o arco que liga o *vértice 0* ao *vértice 1* é percorrido (estes arcos estão graficamente apresentados na figura 4).

2. **Dados**

- O tamanho das ruas;
- O sentido das ruas;
- Poder passar numa rua mais do que uma vez;
- Ter que percorrer todas as ruas no mínimo uma vez.

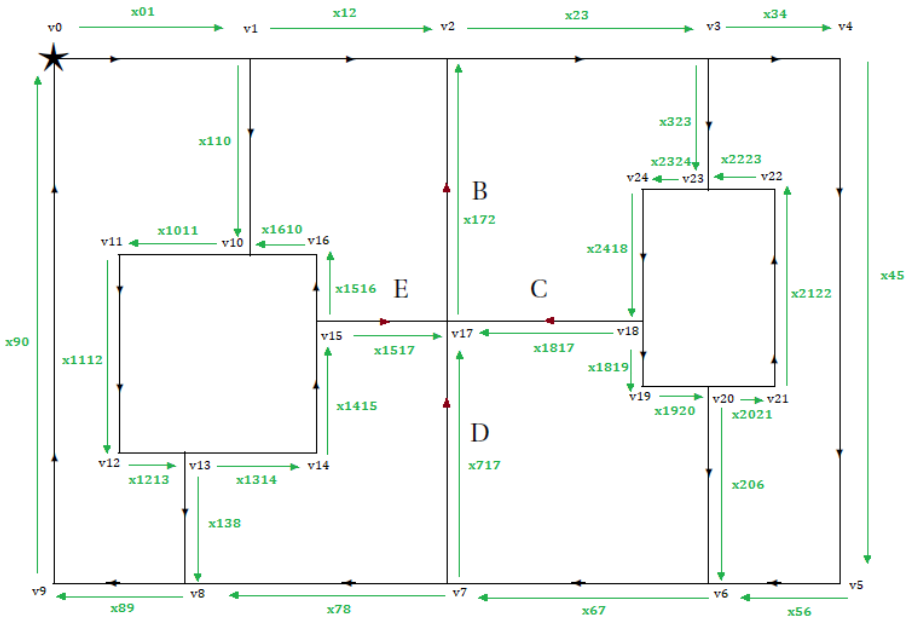


Figura 4: Gráfico com os arcos

2.1.2 Restrições

O numero de vezes que se entra num vértice tem que ser igual ao numero de vezes que se sai do vértice, por isso a soma dos arcos de entrada e de saída tem que ser iguais de modo a garantir que não existe a possibilidade de ficar parado num vértice. Para assegurar que passamos em todos os arcos, todo o x_{ij} tem que ser maior ou igual a 1.

2.1.3 Função objetivo

Como o objetivo visa minimizar a distância percorrida, então é necessário minimizar a soma das distâncias de todos os arcos.

Obtendo então a seguinte função objetivo:

$$\min = 3x_{01} + 3x_{110} + 4x_{23} + 2x_{323} + 2x_{34} + 8x_{45} + 2x_{56} + 4x_{67} + 4x_{78} + 4x_{717} + 2x_{89} + 8x_{90} + 2x_{1011} + 3x_{1112} + 1x_{1213} + 2x_{1314} + 2x_{138} + 2x_{1415} + 2x_{1517} + 1x_{1516} + 1x_{1610} + 4x_{172} + 3x_{1817} + 1x_{1819} + 1x_{1920} + 1x_{2021} + 3x_{206} + 3x_{2122} + 1x_{2223} + 1x_{2324} + 2x_{2418}$$

2.2 Questão 2

Apresente o ficheiro de input (cut-and-paste).

```
/* Objective function */
min: 3 x01 + 3 x110 + 3 x12 + 4 x23 + 2 x323 + 2 x34 + 8 x45 + 2 x56 + 4 x67 + 4 x78
     + 4 x717 + 2 x89 + 8 x90 + 2 x1011 + 3 x1112 + 1 x1213 + 2 x1314 + 2 x138 + 2 x1415
     + 2 x1517 + 1 x1516 + 1 x1610 + 4 x172 + 3 x1817 + 1 x1819 + 1 x1920 + 1 x2021
     + 3 x206 + 3 x2122 + 1 x2223 + 1 x2324 + 2 x2418;
```

Figura 5: Função objetivo.

```
/* RestriÃ§Ãµes */

x110 + x12 -x01 = 0;
x23 - x12 - x172 = 0;
x34 + x323 - x23 = 0;
x45 - x34 = 0;
x56 - x45 = 0;
x67 - x56 - x206 = 0;
x78 + x717 - x67 = 0;
x89 - x78 - x138 = 0;
x90 - x89 = 0;
x1011 - x110 - x1610 = 0;
x1112 - x1011 = 0;
x1213 - x1112 = 0;
x138 + x1314 - x1213 = 0;
x1415 - x1314 = 0;
x1517 + x1516 - x1415 = 0;
x1610 - x1516 = 0;
x172 - x1517 - x717 - x1817 = 0;
x1817 + x1819 - x2418 = 0;
x1920 - x1819 = 0;
x206 + x2021 - x1920 = 0;
x2122 - x2021 = 0;
x2223 - x2122 = 0;
x2324 - x323 - x2223 = 0;
x2418 - x2324 = 0;

x01 >= 1; x110 >= 1; x12 >= 1;
x23 >= 1; x323 >= 1; x34 >= 1;
x45 >= 1; x56 >= 1; x67 >= 1;
x78 >= 1; x717 >= 1; x89 >= 1;
x90 >= 1; x1011 >= 1; x1112 >= 1;
x1213 >= 1; x1314 >= 1; x138 >= 1;
x1415 >= 1; x1517 >= 1; x1516 >= 1;
x1610 >= 1; x172 >= 1; x1817 >= 1;
x1819 >= 1; x1920 >= 1; x2021 >= 1;
x206 >= 1; x2122 >= 1; x2223 >= 1;
x2324 >= 1; x2418 >= 1;
```

Figura 6: Restrições.

```
/* Variable bounds */

int x01,x110,x12,x23,x323,x34,x45,
x56,x67,x78,x717,x89,x90,x1011,
x1112,x1213,x1314,x138,x1415,
x1517,x1516,x1610,x172,x1817,
x1819,x1920,x2021,x206,x2122,
x2223,x2324,x2418;
```

Figura 7: Variáveis.

2.3 Questão 3

Apresente o ficheiro de output produzido pelo programa (cut-and-paste).

Variables	MILP ...	result
	182	182
x01	3	3
x110	2	2
x12	1	1
x23	4	4
x323	3	3
x34	1	1
x45	1	1
x56	1	1
x67	3	3
x78	2	2
x717	1	1
x89	3	3
x90	3	3
x1011	3	3
x1112	3	3
x1213	3	3
x1314	2	2
x138	1	1
x1415	2	2
x1517	1	1
x1516	1	1
x1610	1	1
x172	3	3
x1817	1	1
x1819	3	3
x1920	3	3
x2021	1	1
x206	2	2
x2122	1	1
x2223	1	1
x2324	4	4
x2418	4	4

Figura 8: Resultados.

2.4 Questão 4

Interprete a solução óptima, apresente o percurso correspondente, e calcule o seu custo.

2.4.1 Solução ótima

Analisando o resultado de output produzido pelo programa acima demonstrado podemos concluir que de modo a minimizarmos a distância percorrida devemos percorrer 3 vezes o arco 0 →1, 2 vezes o arco 1 →10, 1 vez o arco 1 →2, 4 vezes o arco 2 →3, 3 vezes o arco 3 →23, 1 vez o arco 3 →4, 1 vez o arco 4 →5, 1 vez o arco 5 →6, 3 vezes o arco 6 →7, 2 vezes o arco 7 →8, 1 vez o arco 7 →17, 3 vezes o arco 8 →9, 3 vezes o arco 9 →0, 3 vezes o arco 10 →11, 3 vezes o arco 11 →12, 3 vezes o arco 12 →13, 2 vezes o arco 13 →14, 1 vez o arco 13 →8, 2 vezes o arco 14 →15, 1 vez o arco 15 →17, 1 vez o arco 15 →16, 1 vez o arco 16 →10, 3 vezes o arco 17 →2, 1 vez o arco 18 →17, 3 vezes o arco 18 →19, 3 vezes o arco 19 →20, 1 vez o arco 20 →21, 2 vezes o arco 20 →6, 1 vez o arco 21 →22, 1 vez o arco 22 →23, 4 vezes o arco 23 →24 e 4 vezes o arco 24 →18.

2.4.2 Percurso ótimo

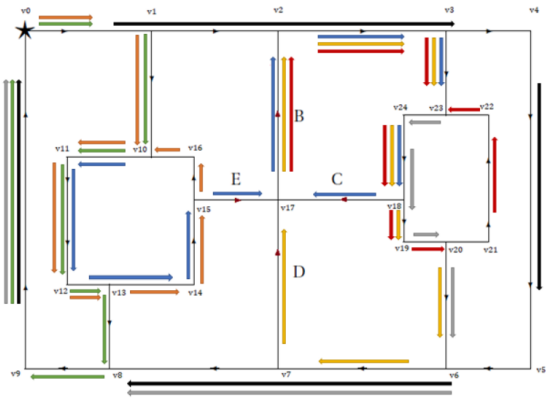


Figura 9: Percuro ótimo.

Inicialmente, o veículo de recolha percorre a parte exterior do mapa, que corresponde ao percurso a preto. Depois inicia o caminho a verde para que o arco 13 →8 seja percorrido, voltando novamente ao ponto inicial. Seguidamente o percurso laranja é percorrido, e dá uma volta ao bloco da esquerda, seguindo se o percurso azul. Este tem como objetivo principal percorrer o arco 18 →17. Para que o arco 7 →27 seja percorrido pelo menos uma vez, tem que haver uma ligação entre o final do percurso azul e outro percurso, desenhado a amarelo. Por último o percurso vermelho é iniciado e dá uma volta total ao bloco da direita, e segue-se do percurso cinzento, que sai do bloco no arco 20 →6, percorrendo a parte exterior do mapa e volta ao ponto inicial, terminando assim a recolha.

2.4.3 Custo

Substituindo os valores ótimos obtidos na solução do *lp_solver* na função objetivo acima apresentada (em 2.1.3) obtemos que o custo é **182** unidades de distância.

2.5 Questão 5

Descreva os procedimentos usados para validar o modelo.

De modo a validar os resultados, tanto na função objetivo como nas restrições apresentadas, foi feita a substituição das variáveis de decisão pelos valores ótimos obtidos através do *lp_solve*. Com isto, o objetivo é provar que as condições apresentadas se verificam e são portanto verdadeiras. Após substituirmos verificamos então que, todas as restrições eram verdadeiras e o resultado da função objetivo era então o custo que tinhamos obtido pelo *lp_solve*. Com o objetivo de assegurar que não havia erros de cálculo, o grupo decidiu criar um pequeno programa escrito em *haskell* que verifica as restrições *(Ver anexo)*.

3 Anexo

```
module Main where

value = 182.00000000

x01 = 3; x110  = 2; x12   = 1;
x23   = 4; x323  = 3; x34   = 1;
x45   = 1; x56   = 1; x67   = 3;
x78   = 2; x717  = 1; x89   = 3;
x90   = 3; x1011 = 3; x1112 = 3;
x1213 = 3; x1314 = 2; x138  = 1;
x1415 = 2; x1517 = 1; x1516 = 1;
x1610 = 1; x172  = 3; x1817 = 1;
x1819 = 3; x1920 = 3; x2021 = 1;
x206  = 2; x2122 = 1; x2223 = 1;
x2324 = 4; x2418 = 4;


min_ = 3*x01 + 3*x110 + 3*x12 + 4*x23 + 2*x323 + 2*x34 + 8*x45 + 2*x56 + 4*x67 +
       4*x78 + 4*x717 + 2*x89 + 8*x90 + 2*x1011 + 3*x1112 + 1*x1213 + 2*x1314 + 2*x138 +
       2*x1415 + 2*x1517 + 1*x1516 + 1*x1610 + 4*x172 + 3*x1817 + 1*x1819 + 1*x1920 +
       1*x2021 + 3*x206 + 3*x2122 + 1*x2223 + 1*x2324 + 2*x2418

l = [x110 + x12 -x01 == 0, x23 - x12 - x172 == 0,x34 + x323 - x23 == 0,x45 - x34 == 0,
    x56 - x45 == 0,x67 - x56 - x206 == 0,x78 + x717 - x67 == 0,x89 - x78 - x138 == 0,
    x90 - x89 == 0,x1011 - x110 - x1610 == 0,x1112 - x1011 == 0,x1213 - x1112 == 0,
    x138 + x1314 - x1213 == 0,x1415 - x1314 == 0,x1517 + x1516 - x1415 == 0,
    x1610 - x1516 == 0,x172 - x1517 - x717 - x1817 == 0,x1817 + x1819 - x2418 == 0,
    x1920 - x1819 == 0,x206 + x2021 - x1920 == 0,x2122 - x2021 == 0,x2223 - x2122 == 0,
    x2324 - x323 - x2223 == 0,x2418 - x2324 == 0]

main = do
    print $ (and l) && (min_ == value)
```

Figura 10: Código de verificação.

4 Conclusão

Com estes procedimentos chegamos à solução ótima pretendida. Sendo assim, para que seja possível passar por todos os arcos de modo a percorrer o minimo de distância possível, é necessário andar uma distância de 182 unidades e mover-se pelo percurso dado como ideal acima. Concluimos isto facilmente com recurso ao lpsolve, sem o mesmo iria-mos demorar relativamente mais tempo e daria mais trabalho, aumentando também a probabilidade de erro.