Modelos Determinísticos de Investigação Operacional

Trabalho prático nº1

16 de outubro de 2019

Grupo nº 11

Filipa Alves dos Santos (A83631)

Hugo André Coelho Cardoso (A85006)

João da Cunha e Costa (A84775)

Válter Ferreira Picas Carvalho (A84464)

Mestrado Integrado em Engenharia Informática

Universidade do Minho

Índice de conteúdos

1. Introdução

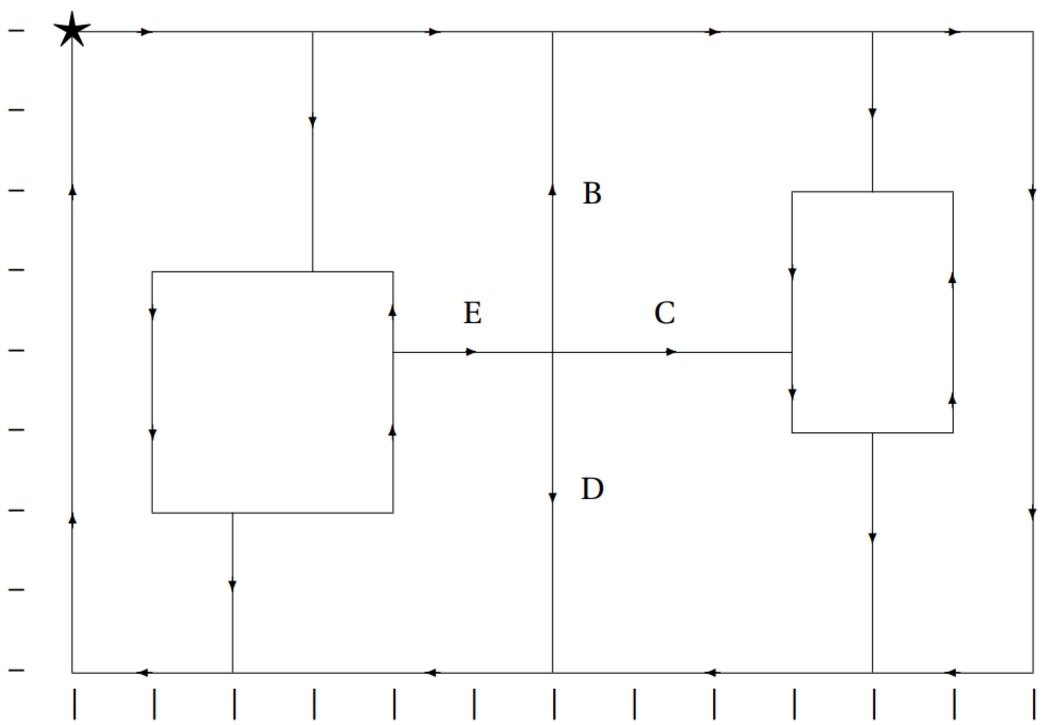
O desafio proposto neste trabalho prático na unidade curricular de Modelos Determinísticos de Investigação Operacional consistiu em minimizar a distância total percorrida num certo circuito. Este tipo de problemas ocorre na vida real, por exemplo na recolha de lixo: os camiões têm de passar por todas as ruas e acabam por repetir algumas delas devido a dificuldades relacionadas com o sentido e disposição de todas as ruas que fazem parte de uma determinada zona.

Assim, para minimizar a distância e satisfazer as condições deste tipo de problemas, criamos várias restrições e uma função objetivo que, através do programa "LPSolve", nos deram uma solução ideal. No resto deste documento vamos explicar o nosso raciocínio, desde a nossa escolha de variáveis de decisão até à verificação do resultado.

2. Desenvolvimento

2.1. Modelo de Programação Linear

Considerando que 85006 é o maior número de aluno dos membros do nosso grupo, determinou-se que o sentido da rua B é a subir, o da D é a descer e o das C e E é para a direita. Abaixo apresentamos a rede da cidade atualizada, com estes sentidos já indicados:



Para solucionar este problema, constatamos primeiro que os dados eram os seguintes: o tamanho e o sentido das ruas, e o facto de termos de percorrer todas as ruas pelo menos uma vez, podendo percorrê-las mais vezes se necessário.

Depois, decidimos definir como vértices de um grafo os locais que representam junções de 2 ou mais ruas com pelo menos 2 sentidos diferentes. Assim, é possível formar um grafo orientado e pesado, pelo que as variáveis de decisão são o número de vezes que cada rua é percorrida. Pretendemos minimizar os seus valores, para obter a menor distância total possível, mas de modo a que todas as ruas (arcos do grafo) sejam percorridas pelo menos uma vez.

Assim, '$\boldsymbol{x_i}$' vai representar o número de vezes que a rua $\boldsymbol{i}$ é percorrida, onde $\boldsymbol{i}$ é natural e $1 \leq \boldsymbol{i} \leq 21$. Na figura seguinte, representamos os vértices a vermelho e os arcos a azul, com os respetivos números associados:

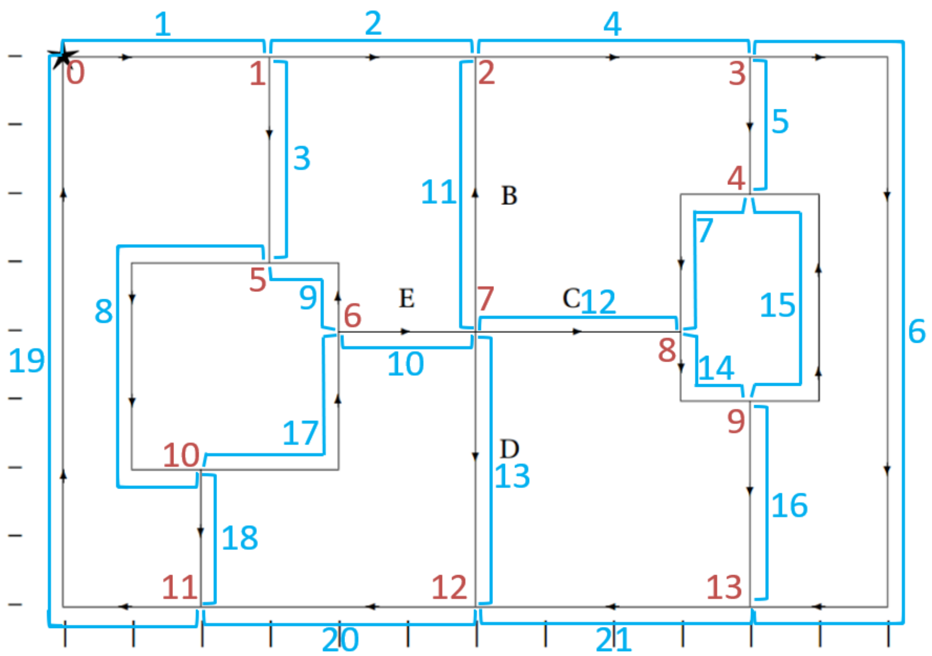


Figura 2 - Mapa com vértices e arcos

No que diz respeito às restrições, o nosso raciocínio baseou-se nos vértices originados pela interseção de ruas. Sabemos que nestes pontos, o número de vezes que o veículo entra, por qualquer rua que tenha como destino esse mesmo vértice, tem de ser igual ao número de vezes que o mesmo sai para quaisquer ruas que originam dele. Isto é, se subtrairmos o número de vezes que passamos nos arcos que saem de um determinado vértice ao número de vezes que atravessamos arcos que o têm como destino, o resultado deverá ser sempre zero (p.e., relativo ao vértice 1, temos a restrição '$x_1 - x_2 - x_3 = 0$', ou seja, no vértice 1 é possível entrar pelo arco x_1 e abandona-se através do arco x_2 ou x_3).

Temos também de assegurar que cada rua é atravessada pelo menos uma vez, isto é:

$$\forall\, 0 \leq i \leq 21: x_i \geq 1, i \in \mathbb{N}$$

Para garantir que, em particular no LPSolve, nos dá uma solução inteira (porque uma rua é atravessada sempre na totalidade), devemos definir as variáveis de decisão como sendo inteiras, como vemos na atribuição de tipos na folha de resolução (ver secção 2.2), no final da *script*.

Já a função objetivo vai envolver também o tamanho de cada arco (calculado através da escala representada no mapa, em que cada tracejado corresponde a uma unidade real, escalada proporcionalmente), sendo que queremos minimizar a distância total e não só o número de vezes que atravessamos cada rua. Assim, a função objetivo vai ter o aspeto da equação abaixo, onde $\boldsymbol{c}$ é o comprimento da rua em unidades reais, proporcionais ao desenho:

$$\min = \sum_{i=0}^{21} c * x_i$$

2.2. Ficheiro Input

```
/* Objective function */
min: 3 x1 + 3 x2 + 3 x3 + 4 x4 + 2 x5 + 12 x6 + 3 x7 + 6 x8 + 2 x9 + 2 x10 + 4
x11 + 3 x12 + 4 x13 + 2 x14 + 5 x15 + 3 x16 + 4 x17 + 2 x18 + 10 x19 + 4 x20 +
4 x21;

/* Variable bounds */
/*V0*/  x19 - x1  = 0;
/*V1*/  x1 - x2 - x3 = 0;
/*V2*/  x2 + x11 - x4 = 0;
/*V3*/  x4 - x5 - x6 = 0;
/*V4*/  x5 + x15 - x7 = 0;
/*V5*/  x3 + x9 - x8 = 0;
/*V6*/  x17 - x9 - x10 = 0;
/*V7*/  x10 - x11 - x12 - x13 = 0;
/*V8*/  x7 + x12 - x14 = 0;
/*V9*/  x14 - x15 - x16 = 0;
/*V10*/ x8 - x17 - x18 = 0;
/*V11*/ x18 + x20 - x19 = 0;
/*V12*/ x13 + x21 - x20 = 0;
/*V13*/ x6 + x16 - x21 = 0;

x1  >= 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;
x14 >= 1;
x15 >= 1;
x16 >= 1;
x17 >= 1;
x18 >= 1;
x19 >= 1;
x20 >= 1;
x21 >= 1;

int x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17,
x18, x19, x20, x21;
```

2.3. Ficheiro Output

Variables	MILP Feasible	Result
	220	220
x1	5	5
x2	1	1
x3	4	4
x4	2	2
x5	1	1
x6	1	1
x7	2	2
x8	5	5
x9	1	1
x10	3	3
x11	1	1
x12	1	1
x13	1	1
x14	3	3
x15	1	1
x16	2	2
x17	4	4
x18	1	1
x19	5	5
x20	4	4
x21	3	3

2.4. Solução Ótima

Através do ficheiro de output, conseguimos retirar a solução ótima: x1 = 5, x2 = 1, x3 = 4, x4 = 2, x5 = 1, x6 = 1, x7 = 2, x8 = 5, x9 = 1, x10 = 3, x11 = 1, x12 = 1, x13 = 1, x14 = 3, x15 = 1, x16 = 2, x17 = 4, x18 = 1, x19 = 5, x20 = 4 e x21 = 3. Com estes resultados, obtemos 5 caminhos (que poderão ser percorridos por qualquer ordem arbitrária), que constituem o percurso total:

- 1 - 2 - 4 - 5 - 7 - 14 - 15 - 7 - 14 - 16 - 21 - 20 - 19
- 1 - 3 - 8 - 17 - 9 - 8 - 18 - 19
- 1 - 3 - 8 - 17 - 10 - 11 - 4 - 6 - 21 - 20 - 19
- 1 - 3 - 8 - 17 - 10 - 12 - 14 - 16 - 21 - 20 - 19
- 1 - 3 - 8 - 17 - 10 - 13 - 20 - 19

(sendo estes números os identificadores dos arcos)

Calculando a distância total através da nossa função objetiva com a solução da tabela, obtemos: $3*5 + 3*1 + 3*4 + 4*2 + 2*1 + 12*1 + 3*2 + 6*5 + 2*1 + 2*3 + 4*1 + 3*1 + 4*1 + 2*3 + 5*1 + 3*2 + 4*4 + 2*1 + 10*5 + 4*4 + 4*3 = 220$, unidades de comprimento reais.

2.5. Validação do Modelo

Validar a nossa solução consiste, maioritariamente, numa série de substituições. Usando os valores das variáveis de decisão na função objetivo, obtemos o resultado adquirido no output do programa. Além disso, verificamos rapidamente que todas as variáveis têm um valor igual ou superior a 1 e, substituindo novamente nas outras restrições (por exemplo, na restrição correspondente ao vértice 1, '$x_1 - x_2 - x_3 = 0$', utilizando os valores obtidos temos: 5 - 1 - 4 = 0 ⇔ 0 = 0), concluímos que o caminho final é válido.

Também conseguimos deduzir, por mera observação da tabela de output, outras conclusões básicas como, por exemplo, o arco 1, o único que sai do vértice 0, tem um número elevado de passagens, pois temos de lá passar para conseguirmos percorrer bastantes outros arcos. De igual modo, os arcos que induzem um ciclo, também terão um número superior de iterações devido à necessidade de passar mais vezes neles para alcançar outros do mesmo ciclo (arcos 8 e 17, por exemplo). Por fim, arcos como o 10 que são basicamente pontos de ligação centrais com outros, também terão um maior número de passagens.

3. Conclusão

Este trabalho incentivou-nos a ter um pensamento lógico de modo a achar as restrições necessárias para resolver o problema e um olhar crítico ao observar os resultados, pois tivemos que perceber se os valores obtidos faziam sentido no contexto do problema e qual o percurso feito de modo a atingir a solução ótima.

Consideramos que foi um trabalho bem sucedido e pretendemos ter resultados semelhantes em futuros trabalhos.