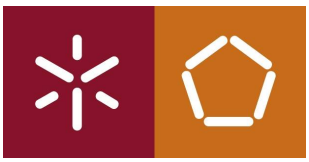

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Modelos Determinísticos de Investigação Operacional

Fluxo de Redes

Etienne Costa (a76089)
João Coutinho (a86272)
Margarida Faria (a71924)
Rui Azevedo (a80789)

29 de Novembro de 2019

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

A realização deste trabalho prático ocorre no âmbito na unidade curricular Modelos Determinísticos de Investigação Operacional e visa desenvolver a capacidade de analisar sistemas complexos, de criar modelos para os descrever, de obter soluções para esses modelos utilizando programas computacionais adequados validando e interpretando as soluções obtidas de modo a elaborar recomendações para o sistema em análise. Neste caso em concreto, pretende-se modelar um conjunto de circuitos em que todos os arcos de um grafo são percorridos, pelo menos uma vez, minimizando a distância total percorrida.

Numa primeira fase do trabalho, serão definidos o conjunto de circuitos óptimos a serem percorridos pelo veículo. Na segunda parte do trabalho irá ser feita uma análise diferente do problema, não tendo em conta os circuitos óptimos a percorrer mas sim os caminhos óptimos de reposicionamento do veículo na rede.

2 Trabalho Prático I

Vários problemas práticos em Investigação Operacional podem ser expressos como problemas de programação linear e este problema não foge à regra. Sendo assim, optou-se por sintetizar o problema em cinco pontos distintos.

2.1 Definição do problema

Um problema comum no âmbito da Investigação Operacional é o fluxo de redes. É um cenário corrente em empresas de distribuição que pretendem entregar produtos aos seus clientes de modo a minimizar os custos da viagem, garantindo que todos os seus clientes recebem as suas encomendas.

O trabalho a desenvolver trata-se de um problema de fluxo de redes onde existe um camião do lixo que parte de um ponto inicial e tem que percorrer todas as ruas da cidade de modo a recolher todo o lixo das ruas acabando o trajecto no ponto de partida. É de ressalvar que o camião deverá passar em todas as ruas pelo menos uma vez e que estas têm um comprimento inteiro, proporcional à dimensão do traço em centímetros.

A baixo é apresentado o mapa da cidade a estudar. Note-se que todas as ruas são de sentido único, sendo muito provável que o camião tenha que passar mais do que uma vez na mesma rua, de maneira a conseguir voltar sempre ao ponto inicial (vértice marcado com a estrela).

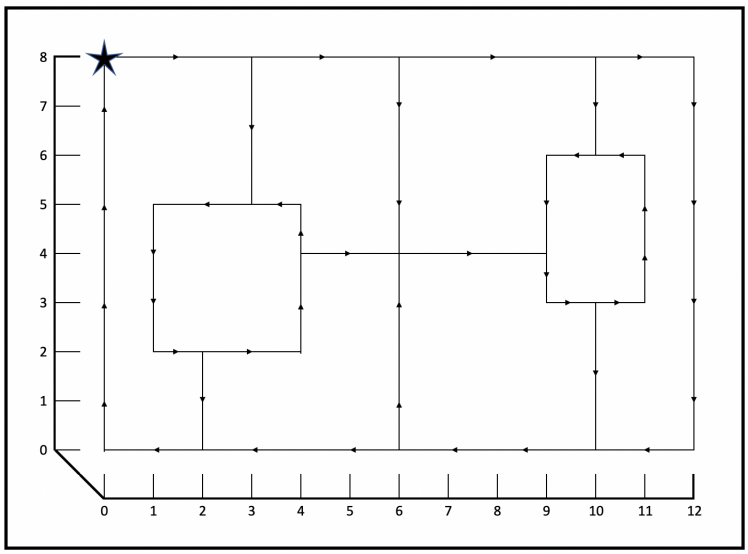

Figura 1: Mapa da cidade

2.2 Análise do problema

A solução para este problema passa por descobrir o circuito ou conjunto de circuitos possíveis que minimizam o custo do percurso. Naturalmente, a solução preferível seria passar uma única vez em cada rua começando e acabando no mesmo ponto.

Após uma breve análise ao mapa da cidade, é visível que existem ruas que vão ter que ser percorridas mais do que uma vez. Na *figura 2*, tem-se um exemplo em que a *rua 3* vai ter que ser percorrida, no mínimo, duas vezes, uma vez que a *rua 1* e a *rua 2* têm ambas que ser percorridas e a única rua que permite a saída destas mesmas é a *rua 3*.

Ao longo do mapa encontram-se várias situações semelhantes a esta, daí poder-se concluir que a solução preferível, que seria percorrer todas as ruas uma única vez, é impossível, devido à topologia das ruas e à restrição de que todas as ruas têm que ser visitadas. Da mesma maneira, é possível identificar ruas em que é suficiente visitá-las uma única vez.

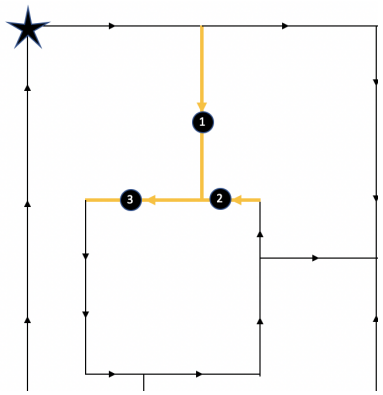

Figura 2: Sub-Mapa da cidade

Será também desejável abstrair o problema e representar o mapa da cidade como um grafo orientado (digrafo)[2]:

$$G = (V,A,v_i)$$

onde V representa o conjunto de todos os vértices, ou seja, os locais onde se troca de rua, A é o conjunto ordenado dos vértices (arcos), que, neste caso representará o conjunto das ruas da cidade e $v_i \in V$ representa o ponto de partida. Para o problema ter solução, é imperativo que o grafo seja fortemente ligado, isto é, a partir de cada vértice existir pelo menos um caminho para qualquer outro vértice. Como a topologia da rede é pequena, verifica-se facilmente que tal regra é obedecida. Numa topologia de rede mais complexa, não era tão perceptível esta propriedade do grafo, sendo que seria necessário usar um algoritmo para provar o fecho transitivo de cada vértice (algoritmo de *Floyd-Warshall*, por exemplo).

Nesta primeira fase de análise, pode-se retirar no mínimo três observações importantes que ajudarão a modelar o problema em questão:

- É impossível, devido à topologia da rede, passar uma única vez em todas as ruas.
- A solução do problema irá ser um conjunto de sub-circuitos, isto é, vários percursos que saem do ponto inicial e voltam ao mesmo, passando num sub-conjunto dos arcos do grafo.
- O número de vezes que se passa no vértice inicial determina o número de sub-circuitos necessários para resolver o problema.

2.3 Construção do modelo

A construção do modelo consiste na especificação de expressões quantitativas para o objectivo e **restrições** do problema, em função das suas **variáveis de decisão**. Dentro da construção do modelo podem distinguir-se três componentes básicas:

1. **Variáveis de decisão e Parâmetros**
   (a) Variáveis de decisão: incógnitas do problema. Representam os níveis das actividades[1].
   (b) Parâmetros: variáveis controladas do sistema, representando, por exemplo, os níveis de recursos disponíveis[1].
2. **Condições, Constrangimentos ou Restrições**
   (a) Restrições: explicitam as regras do funcionamento do sistema. Uma solução que obedeça ao conjunto de condições designa-se por **solução válida**[1].
3. **Função Objectivo**
   (a) Define a medida de eficiência do sistema em função das variáveis de decisão. Permite determinar qual a melhor solução de um conjunto de soluções válidas[1].

2.4 Solução do modelo

A solução do modelo consiste na aplicação de algoritmos existentes ou no desenvolvimento de novos algoritmos para obtenção da solução ótima do modelo.

Visto que este problema consiste em minimizar a distância total percorrida atravessando todos os arcos, optou-se por utilizar a conservação de fluxo e as restrições de não negatividade de modo a obter o conjunto de soluções válidas. A maneira mais simples de tratar problemas do género é através da Programação Linear.

A Programação Linear é um método para atingir o melhor resultado num modelo matemático (quer ele seja de maximização ou minimização) cujas restrições são funções lineares.

Após a modelação do trabalho, irá ser usada a ferramenta *lp_solve* para calcular a solução óptima do problema. Esta ferramenta baseia-se no algoritmo *Simplex*, desenvolvido por *George Dantzig*, para obter a solução óptima de um modelo de programação linear.

2.4.1 Modelo matemático

Após uma análise detalhada do problema e uma vez definidas as ferramentas a usar, bem como a metodologia a aplicar, irá-se definir o modelo matemático que modela o grafo em questão.

2.4.2 Variáveis de decisão

A resolução do problema parte por descobrir qual o caminho com menos custo percorrendo todos os arcos do grafo. Desta maneira, o custo do percurso vai depender do número de vezes que se passa em cada arco.

Dado isto, este problema tem apenas uma variável de decisão, $x_{i,j}$, que representa o número de vezes que se passa no arco que liga o vértice i ao vértice j.

2.4.3 Função objectivo

A função objectivo vai depender então do número de vezes que se passa em cada arco e do custo desses mesmos arcos. Para além disso, é necessário minimizar esta função de maneira a ter o menor custo possível.

Podemos então definir a função objectivo como:

$$\min \sum_{(i,j)\in A} c_{i,j} x_{i,j}$$

$x_{i,j}$: número de vezes que se passa no arco que liga os vértices i e j
$c_{i,j}$: custo do arco, em centímetros, que liga os vértices i e j
A : conjunto de todos os arcos da rede

2.4.4 Restrições

As restrições implementadas ditam as regras para o correto funcionamento do sistema. Com base nisso, dividiu-se as restrições em três grupos:

- **Restrições de conservação de fluxo**: são as restrições que permitem formar caminhos. Para definir um circuito ou conjunto de sub-circuitos temos que garantir que todos os vértices são balanceados, isto é, o número de vezes que se entra num vértice tem que ser igual à soma do número de vezes que se sai desse mesmo vértice.

$$\sum_{(k,i)\in A} x_{k,i} = \sum_{(i,j)\in A} x_{i,j}, \quad \forall i \in V$$

- **Restrições de não negatividade**: são as restrições que garantem que o conjunto de soluções válidas sejam positivas e, neste caso, maiores que 0, de maneira a garantir que todos os arcos são visitados pelo menos uma vez.

$$x_{i,j} \geq 1$$

- **Restrições de integralidade**: são as restrições que definem o domínio do conjunto de soluções válidas. Neste caso, o valor de cada arco tem que ser um valor inteiro positivo

$$x_{i,j} \in \mathbb{N}^+$$

2.5 Modelo final

De seguida, é apresentado o modelo matemático final.

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} c_{i,j} x_{i,j} \\ \text{s.a} \quad & x_{i,j} \geq 1, && x_{i,j} \in \mathbb{N}^+ \\ & \sum_{(k,i)\in A} x_{k,i} = \sum_{(i,j)\in A} x_{i,j}, && \forall i \in V \end{aligned}$$

2.5.1 Ficheiro de input

É necessário agora transcrever o modelo matemático para a linguagem que o *lp_solve* reconhece. De seguida, é apresentado o ficheiro de *input* definido de acordo com o modelo desenvolvido.

```
1
2  /* Funcao Objectivo */
3
4  min: 3*x0_1   + 3*x1_2   + 3*x1_10  + 4*x2_3   + 4*x2_24  + 2*x3_4   +
5       2*x3_17  + 8*x4_5   + 2*x5_6   + 4*x6_7   + 4*x7_24  + 4*x7_8   +
6       2*x8_9   + 8*x9_0   + 2*x10_11 + 3*x11_12 + 1*x12_13 + 2*x13_8  +
7       2*x13_14 + 2*x14_15 + 1*x15_16 + 1*x16_10 + 2*x15_24 + 3*x24_19 +
8       1*x19_20 + 1*x20_21 + 3*x21_6  + 1*x21_22 + 3*x22_23 + 1*x23_17 +
9       1*x17_18 + 2*x18_19 ;
10
11  /* Restricoes de controlo de fluxo */
12
13 x0_1 = x1_10 + x1_2;         x1_2 = x2_24 + x2_3;
14 x2_3 = x3_17 + x3_4;         x3_4 = x4_5;
15 x4_5 = x5_6;                 x5_6 + x21_6 = x6_7;
16 x6_7 = x7_24 + x7_8;         x7_8 + x13_8 = x8_9;
17 x8_9 = x9_0;                 x1_10 + x16_10 = x10_11;
18 x10_11 = x11_12;             x11_12 = x12_13;
19 x12_13 = x13_14 + x13_8;     x13_14 = x14_15;
20 x0_1 = x9_0;                 x14_15 = x15_24 + x15_16;
21 x15_16 = x16_10;             x3_17 + x23_17 = x17_18;
22 x17_18 = x18_19;             x18_19 + x24_19 = x19_20;
23 x19_20 = x20_21;             x20_21 = x21_22 + x21_6;
24 x21_22 = x22_23;             x22_23 = x23_17;
25 x2_24 + x15_24 + x7_24 = x24_19;
26
27 /* Restricoes de nao negatividade */
28 x0_1 >= 1;       x1_2 >= 1;       x2_3 >= 1;
29 x3_4 >= 1;       x4_5 >= 1;       x5_6 >= 1;
30 x6_7 >= 1;       x7_8 >= 1;       x8_9 >= 1;
31 x9_0 >= 1;       x1_10 >= 1;      x10_11 >= 1;
32 x11_12 >= 1;     x12_13 >= 1;     x13_14 >= 1;
33 x14_15 >= 1;     x15_16 >= 1;     x16_10 >= 1;
34 x13_8 >= 1;      x3_17 >= 1;      x17_18 >= 1;
35 x18_19 >= 1;     x19_20 >= 1;     x20_21 >= 1;
36 x21_22 >= 1;     x22_23 >= 1;     x23_17 >= 1;
37 x21_6 >= 1;      x15_24 >= 1;     x24_19 >= 1;
38 x2_24 >= 1;      x7_24 >= 1;
39
40 /* Restricoes de integralidade */
41 int  x0_1   , x1_2.  , x1_10  , x2_3   , x2_24  , x3_4   , x3_17  ,
42      x5_6   , x6_7   , x7_24  , x7_8.  , x8_9   , x9_0   , x10_11 ,
43      x12_13 , x13_8  , x13_14 , x14_15 , x15_16 , x16_10 , x15_24 ,
44      x24_19 , x19_20 , x20_21 , x21_6  , x21_22 , x22_23 , x23_17 ,
45      x17_18 , x18_19 , x4_5   , x11_12;
```

Listing 1: *Script* que transcreve o modelo desenvolvido

2.5.2 Ficheiro de *output*

Tendo como base o método *Simplex*, o *lp_solve* calculou o valor das variáveis de decisão que minimizam o custo do percurso no grafo. De seguida, apresenta-se o conjunto de soluções válidas.

```
Valor da funcao objectivo: 220.00

Valor atual das variaveis:
x0_1    = 5       x1_2    = 3       x1_10   = 2
x2_3    = 2       x2_24   = 1       x3_4    = 1
x3_17   = 1       x4_5    = 1       x5_6    = 1
x6_7    = 5       x7_24   = 1       x7_8    = 4
x8_9    = 5       x9_0    = 5       x10_11  = 3
x11_12  = 3       x12_13  = 3       x13_8   = 1
x13_14  = 2       x14_15  = 2       x15_16  = 1
x16_10  = 1       x15_24  = 1       x24_19  = 3
x19_20  = 5       x20_21  = 5       x21_6   = 4
x21_22  = 1       x22_23  = 1       x23_17  = 1
x17_18  = 2       x18_19  = 2
```

Listing 2: *Output* produzido pelo *lp_solve*

2.6 Validação do modelo

Uma vez calculada a solução do problema, pode-se representar o grafo com os valores obtidos pelo *lp_solve*.

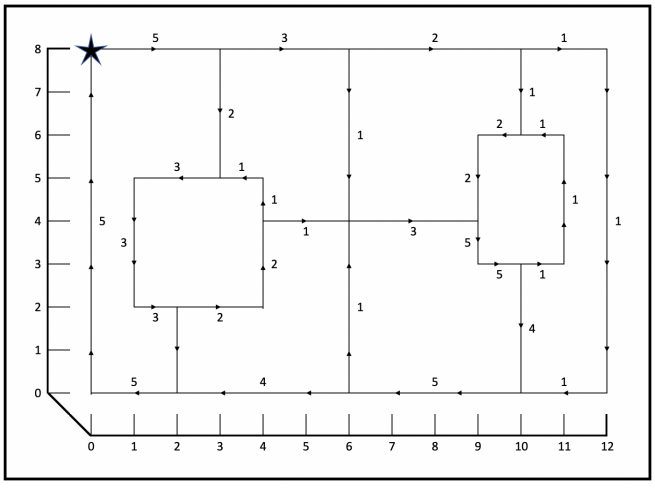

Figura 3: Mapa com o conjunto de soluções

É necessário agora interpretar os resultados e definir o conjunto dos sub-circuitos que minimizam o custo total do trajecto. Este processo passa por ir traçando arcos, decrementando o seu valor, até atingir o valor de 0. Quando é atingido este limite, o arco em questão já não necessita de ser visitado.

De seguida, é apresentado o conjunto de sub-circuitos obtidos, uma vez feita a análise dos resultados. Note-se que a solução apresentada não é única, isto é, é possível apresentar um conjunto diferente de soluções do que aquele apresentado a baixo obtendo, mesmo assim, o custo mínimo possível.

Os arcos coloridos representam o percurso a percorrer. Existem três cores possíveis para estes arcos e têm o seguinte significado:

- Laranja : Arcos percorridos apenas uma vez
- Azul : Arcos percorridos duas vezes
- Vermelho : Arcos percorridos três vezes

Os circuitos *1, 2* e *3* representados nas *figuras 4* e *5* são circuitos acíclicos, isto é, é possível traçar um percurso que parta e regresse ao vértice inicial passando apenas uma vez nos arcos desse mesmo percurso.

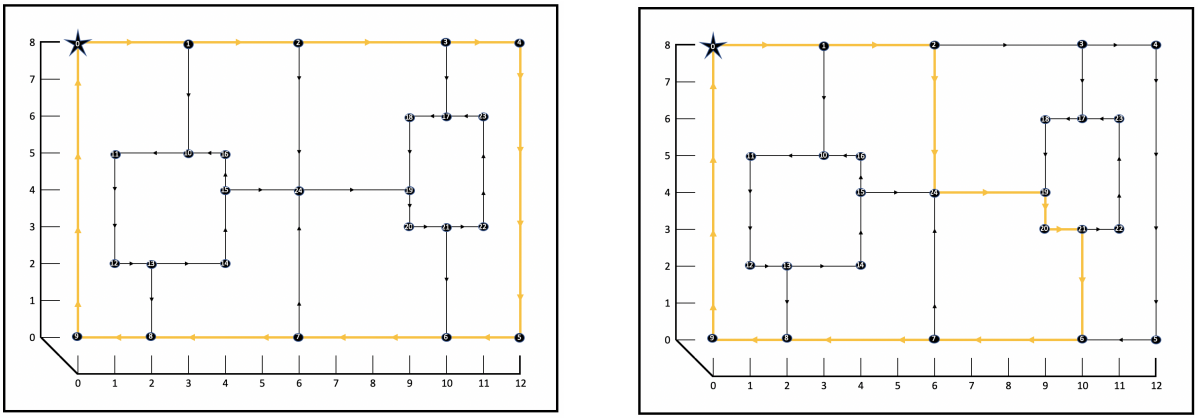

Figura 4: Circuito 1 e Circuito 2

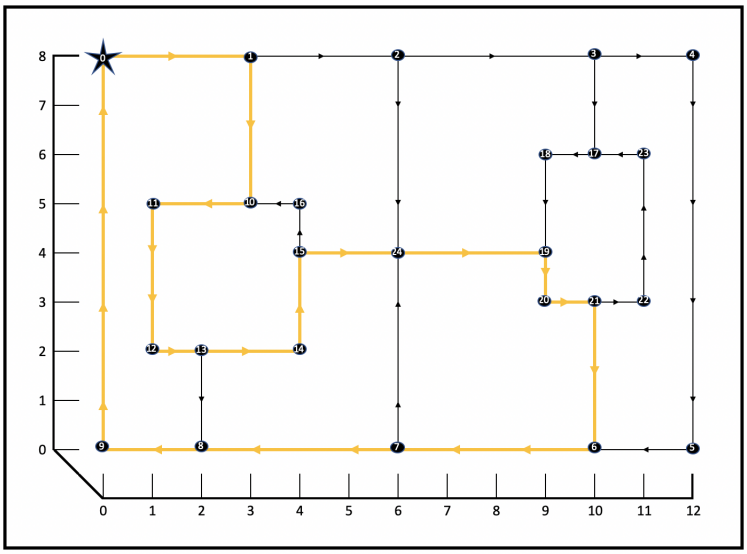

Figura 5: Circuito 3

Na *figura 6* estão representados circuitos cíclicos, isto é, existem arcos que têm que ser percorridas mais do que uma vez de modo a conseguir-se voltar ao vértice inicial.

No *Circuito 4* existem duas maneiras distintas de percorrer o grafo, e são as seguintes:

- **Hipótese 1**

$$x_{0,1} \Longrightarrow x_{1,2} \Longrightarrow x_{2,3} \Longrightarrow x_{3,17} \Longrightarrow x_{17,18} \Longrightarrow x_{18,19} \Longrightarrow x_{19,20} \Longrightarrow x_{20,21} \Longrightarrow x_{21,22} \Longrightarrow x_{22,23} \Longrightarrow x_{23,17} \Longrightarrow x_{17,18} \Longrightarrow x_{18,19} \Longrightarrow x_{19,20} \Longrightarrow x_{20,21} \Longrightarrow x_{21,6} \Longrightarrow x_{6,7} \Longrightarrow x_{7,24} \Longrightarrow x_{24,19} \Longrightarrow x_{19,20} \Longrightarrow x_{20,21} \Longrightarrow x_{21,6} \Longrightarrow x_{20,21} \Longrightarrow x_{6,7} \Longrightarrow x_{7,8} \Longrightarrow x_{8,9} \Longrightarrow x_{9,0}$$

- **Hipótese 2**

$$x_{0,1} \Longrightarrow x_{1,2} \Longrightarrow x_{2,3} \Longrightarrow x_{3,17} \Longrightarrow x_{17,18} \Longrightarrow x_{18,19} \Longrightarrow x_{19,20}x_{20,21} \Longrightarrow x_{21,6} \Longrightarrow x_{6,7} \Longrightarrow x_{7,24} \Longrightarrow x_{24,19} \Longrightarrow x_{19,20} \Longrightarrow x_{20,21} \Longrightarrow x_{20,21} \Longrightarrow x_{21,22} \Longrightarrow x_{22,23} \Longrightarrow x_{23,17} \Longrightarrow x_{17,18} \Longrightarrow x_{18,19} \Longrightarrow x_{19,20} \Longrightarrow x_{20,21} \Longrightarrow x_{21,6} \Longrightarrow x_{6,7} \Longrightarrow x_{7,8} \Longrightarrow x_{8,9} \Longrightarrow x_{9,0}$$

O *Circuito 5*, embora tenha três arcos que têm que ser percorridas mais do que uma vez, só existe um trajecto admissível:

$$x_{0,1} \Longrightarrow x_{1,10} \Longrightarrow x_{10,11} \Longrightarrow x_{11,12} \Longrightarrow x_{12,13} \Longrightarrow x_{13,14} \Longrightarrow x_{14,15} \Longrightarrow x_{15,16} \Longrightarrow x_{16,10} \Longrightarrow x_{10,11} \Longrightarrow x_{11,12} \Longrightarrow x_{12,13} \Longrightarrow x_{13,8} \Longrightarrow x_{8,9} \Longrightarrow x_{9,0}$$

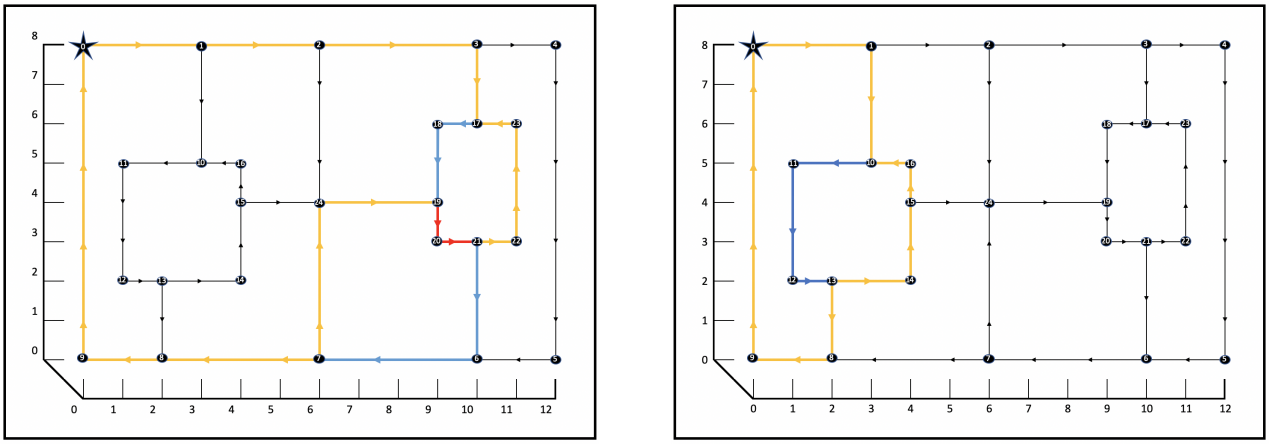

Figura 6: Circuito 4 e Circuito 5

Substituindo agora as variáveis de decisão na função objectivo, obtém-se o custo mínimo do percurso. O custo mínimo total é igual à soma do custo dos sub-circuitos.

Sendo c_x o custo do sub-circuito x, pode-se formular que:

$$\begin{aligned} c_{Total} &= c_{c1} + c_{c2} + c_{c3} + c_{c4} + c_{c5} \\ &= 40 + 36 + 44 + 64 + 36 \\ &= 220cm \end{aligned}$$

Tabela 1: Tabela de custos

Circuitos	Custo(cm)
1	40
2	36
3	44
4	64
5	36
Total	220

Pode-se concluir que a solução obtida para o problema é uma solução admissível pois obedece a todas as restrições definidas. Na *figura 3* observa-se que, em cada vértice, o valor de entrada é igual à soma dos valores de saída, garantindo, desta forma, a conservação do fluxo da rede. Para além disso, sendo todos os valores obtidos positivos e inteiros, pode-se afirmar que todos os arcos são percorridas pelo menos uma vez. A solução obtida é também óptima para o modelo definido pois foi usado o método *Simplex*, através da ferramenta *lp_solve*, para a calcular.

Como foi mencionado anteriormente, a solução para este problema não é única, sendo possível traçar trajectos diferentes mas que no fim resultam no mesmo valor na função objectivo.

Equiparando com a situação do mundo real dos camiões do lixo, o condutor do camião poderia escolher a ordem pela qual os sub-circuitos são percorridos, uma vez que a distância final percorrida seria a mesma.

2.7 Discussão

Nesta secção vão-se apresentar as dificuldades encontradas no desenvolvimento do trabalho, assim como uma abordagem diferente para a modelação do grafo.

Inicialmente, acreditou-se que a solução passaria por modelar uma árvore de custo mínimo (*MST - Minimum Spam Tree*), no entanto, a propriedade que define que uma árvore é um sub-grafo acíclico não permite, para este problema em específico, garantir que todos os arcos são visitadas pelo menos uma vez.

A abordagem seguinte foi pensar em definir a variável de decisão como uma variável binária, isto é, o valor de 0 representava um arco não ter sido percorrida e o valor de 1 o contrário. O resultado desta abordagem vem com um problema muito grande, pois, com isto, uma vez que todos os arcos têm que ser percorridas pelo menos uma vez, o valor de todas as variáveis de decisão seria 1, ou seja, só se saberia que todos os arcos tinham sido percorridos, não conseguindo definir os caminhos que foram usados.

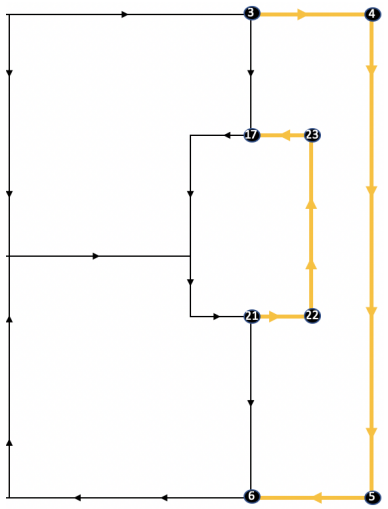

Figura 7: Sub-Mapa da cidade

Por fim, acabou-se por definir a variável de decisão como o número de vezes que se passa num determinado arco, conseguindo assim modelar correctamente o problema.

Era possível otimizar o modelo desenvolvido para este caso em específico. Uma vez que existem casos em que só existe um caminho possível entre dois vértices não adjacentes, podia-se assumir que o conjunto de arcos que percorrem esses mesmos vértices formam apenas um arco. Por exemplo, na *figura 7*, encontram-se duas situações representativas do que foi dito, podemos sintetizar o conjunto de vértices $\{v_{3,4}, v_{4,5}, v_{5,6}\}$ em apenas $v_{3,6}$, assim como o conjunto $\{v_{21,22}, v_{22,23}, v_{23,17}\}$ em $v_{21,17}$. Esta abordagem tem as suas vantagens e desvantagens. Por um lado, conseguia-se reduzir o número de restrições para este problema, tornando o processo de cálculo da solução mais eficiente. Por outro lado, perdia-se o facto do modelo matemático desenvolvido não ser adaptável a uma rede diferente. Note-se que o modelo desenvolvido é genérico para qualquer grafo orientado, dado que se trabalha com o balanço de cada vértice do grafo, não estando dependente da sua topologia.

3 Trabalho Prático II (Parte I)

3.1 Definição do problema

Os *softwares* de resolução de problemas de transporte em rede, normalmente, funcionam com limites superiores. Uma vez que a solução apresentada anteriormente foi desenvolvida com limites inferiores, sendo estes limites maiores ou iguais a um, é necessário reformular o problema de maneira a acrescentar limites superiores para os valores das variáveis de decisão e forçar os limites inferiores a terem valor de zero.

Aplicando a mudança de variável $y_{i,j} = x_{i,j} - l_{i,j}, \forall (i,j) \in A$, em que $l_{i,j}$ é o limite inferior do fluxo no arco (i,j), obtém-se então uma nova instância do problema. Com isto, é necessário reformular também a interpretação dos resultados obtidos com esta nova instância do problema uma vez que as estruturas das soluções são diferentes das anteriormente apresentadas.

3.2 Formulação do problema

A mudança de variável $y_{i,j} = x_{i,j} - l_{i,j}$ faz com que todo o problema tenha que ser reformulado. Com base nisto, enumera-se de seguida os diferentes aspectos do modelo que irão sofrer alterações.

3.2.1 Variáveis de decisão

A variável $y_{i,j}$ passa a ter um novo significado no contexto do problema. Enquanto que na formulação anterior, a variável $x_{i,j}$ representava o número de vezes que uma aresta era percorrida, nesta nova instância do problema tem-se que $y_{i,j}$ é o número de vezes que o veículo tem que ser reposicionado na aresta que liga os vértices i e j.

O percurso óptimo, que seria percorrer todas as arestas uma única vez, implicava a seguinte solução:

$$\forall (i,j) \in A, y_{i,j} = 0$$

Este conjunto de soluções significa que o veículo não sofreu reposicionamento em nenhuma aresta.

3.2.2 Função Objectivo

Aplicando a mudança de variável à função objectivo obtemos a seguinte expressão:

$$\min \quad \sum_{(i,j)\in A} c_{i,j}(y_{i,j}+1), \quad \forall (i,j) \in A$$

Pode-se observar que, caso o veículo não seja reposto numa determinada aresta, ou seja, o fluxo da mesma tem valor de zero, essa aresta terá na mesma influência na função objectivo:

$$y_{i,j} = 0 \rightarrow (z = c_{i,j}(0+1) \Leftrightarrow z = c_{i,j})$$

É possível ainda simplificar a expressão da seguinte maneira:

$$\min \quad \sum_{(i,j)\in A} c_{i,j} + \sum_{(i,j)\in A} c_{i,j}y_{i,j}, \quad \forall (i,j) \in A$$

Com a simplificação da expressão da função objectivo, há uma propriedade que é salientada, a solução é partida em duas componentes. A primeira componente é a que garante que todas as arestas têm que ser visitadas, daí o primeiro somatório ser independente das variáveis de decisão, representando o custo total da rede. A segunda componente diz respeito ao reposicionamento do veículo nas arestas, ou seja, o segundo somatório só irá acrescentar à função objcetivo o valor das arestas que irão ser revisitadas.

3.2.3 Restrições

As restrições do modelo são reformuladas da seguinte maneira:

$$-\sum_{(k,i)\in A} (y_{k,i}+1) + \sum_{(i,j)\in A} (y_{i,j}+1) = 0, \qquad \forall i \in V$$

$$1 \leq y_{i,j} + 1 \leq 100, \qquad y_{i,j} \in \mathbb{N}$$

Mais uma vez, é possível simplificar as restrições obtendo a seguinte expressão:

$$
\begin{aligned}
-\sum_{(k,i)\in A} y_{k,i} + \sum_{(i,j)\in A} y_{i,j} = b_i, \qquad & \forall i \in V \\
0 \leq y_{i,j} \leq 99, \qquad & y_{i,j} \in \mathbb{N}
\end{aligned}
$$

Com a simplificação do modelo, é acrescentada ao mesmo uma variável b_i que representa o fator de balanceamento de um vértice, *i.e*, sendo E_i o conjunto das arestas de entrada do vértice i e S_i o conjunto de arestas de saída do vértice i, então $b_i = \#S_i - \#E_i$, $\forall i \in V$.

Pode-se então afirmar que, com a mudança de variável passam a existir:

- Vértices de excesso: vértices cujo grau de entrada é superior ao grau de saída, ou seja, $b_{i,j} > 0$.
- Vértices de defeito: vértices cujo grau de entrada é menor ao grau de saída, ou seja, $b_{i,j} < 0$.
- Vértices balanceados: vértices cujo grau de entrada é igual ao grau de saída, ou seja, $b_{i,j} = 0$.

A rede resultante da transformação do modelo é apresentada de seguida. Note-se que os vértices balanceados podem ser eliminados mantendo na rede apenas vértices de excesso e defeito. Isto deve-se ao facto do valor do fluxo se manter igual nessas arestas.

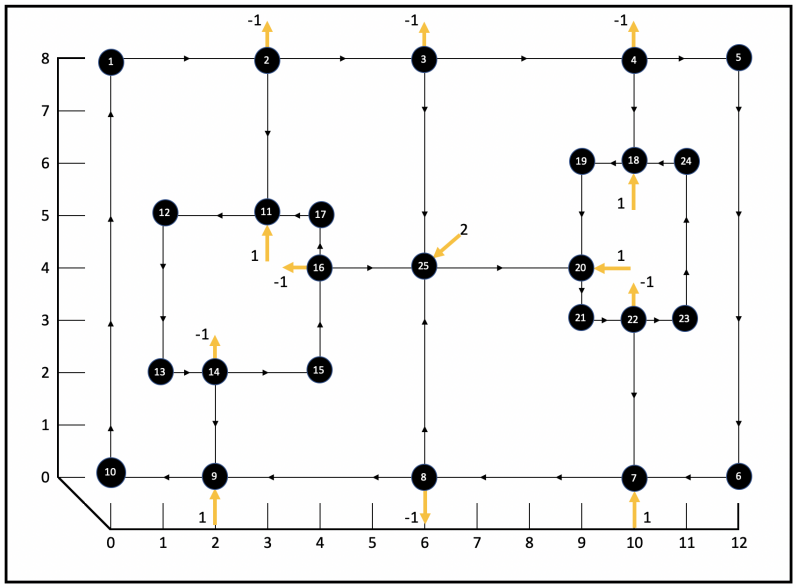

Figura 8: Rede com ofertas e procuras

3.2.4 Estrutura das soluções

Enquanto as soluções do primeiro modelo definido permitem traçar circuitos de modo a garantir que todas as arestas são visitadas, as soluções deste modelo têm uma estrutura diferente. Neste caso, o fluxo das arestas representa o número de vezes que o veículo é reposicionado numa determinada aresta.

O **teorema da decomposição de fluxo** garante que qualquer fluxo numa rede consegue ser decomposto num conjunto de caminhos e ciclos tal que:

- Cada caminho direccionado com fluxo positivo, liga um vértice de excesso a um vértice de defeito.
- No máximo, existem $n + m$ caminhos ou ciclos, onde n é o número de vértices e m o número de vértices.
- O custo de um caminho é dado pela soma dos custos de cada aresta contida no caminho.

Efectivamente, a estrutura da solução para este modelo permite traçar caminhos que vão voltar a ser revisitados pelo veículo deixando de parte caminhos que apenas tenham que ser visitados apenas uma vez. De facto, é esta condição que não permite, de uma forma tão directa, traçar os diferentes circuitos.

3.3 Modelo final

De seguida, é apresentado o modelo final resultante da mudança de variável.

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} c_{i,j} + \sum_{(i,j)\in A} c_{i,j} y_{i,j} \\ \text{s.a} \quad & -\sum_{(k,i)\in A} y_{k,i} + \sum_{(i,j)\in A} y_{i,j} = b_i, && \forall i \in V \\ & 0 \leq y_{i,j} \leq 99, && y_{i,j} \in \mathbb{N} \end{aligned}$$

3.3.1 Ficheiro de *Input* do *lp_solve*

```
/* Funcao Objetivo */
    min: 85 + 3*y0_1   + 3*y1_2   + 3*y1_10  + 4*y2_3   + 4*y2_24  +
              2*y3_4   + 2*y3_17  + 8*y4_5   + 2*y5_6   + 4*y6_7   +
              4*y7_24  + 4*y7_8   + 2*y8_9   + 8*y9_0   + 2*y10_11 +
              3*y11_12 + 1*y12_13 + 2*y13_8  + 2*y13_14 + 2*y14_15 +
              1*y15_16 + 1*y16_10 + 2*y15_24 + 3*y24_19 + 1*y19_20 +
              1*y20_21 + 3*y21_6  + 1*y21_22 + 3*y22_23 + 1*y23_17 +
              1*y17_18 + 2*y18_19;

 /* Restricoes */
-y0_1 + y1_10 + y1_2 = -1;                -y2_3 + y3_17 + y3_4 = -1;
-y4_5 + y5_6 = 0;                         -y6_7 + y7_24 + y7_8 =-1;
-y8_9 + y9_0 = 0;                         -y10_11 + y11_12 =0;
-y12_13 + y12_14 + y13_8 = -1;            -y9_0 + y0_1 = 0;
-y15_16 + y16_10 =0;                      -y17_18 + y18_19 = 0;
-y19_20 + y20_21 = 0;                     -y21_22 + y22_23 =0;
-y2_24 - y15_24 -y7_24 + y24_19 = 2;      -y1_2 + y2_24 + y2_3 = -1;
-y3_4 + y4_5 =0;                          -y5_6 - y21_6 + y6_7 = 1;
-y7_8 -y13_8 + y8_9=1;                    -y1_10 - y16_10 + y10_11 = 1;
-y11_12 + y12_13 = 0;                     -y13_14 + y14_15 = 0;
-y14_15 + y15_24 + y15_16 = -1;           -y3_17 -y23_17 + y17_18 =1 ;
-y18_19 - y24_19 + y19_20 =1;             -y20_21 + y21_22 + y21_6 = -1;
-y22_23 + y23_17 =0;

/* Restricoes de nao negatividade */
y0_1 >= 0;       y1_2 >= 0;       y2_3 >= 0;
y3_4 >= 0;       y4_5 >= 0;       y5_6 >= 0;
y6_7 >= 0;       y7_8 >= 0;       y8_9 >= 0;
y9_0 >= 0;       y1_10 >= 0;      y10_11 >= 0;
y11_12 >= 0;     y12_13 >= 0;     y13_14 >= 0;
y14_15 >= 0;     y15_16 >= 0;     y16_10 >= 0;
y13_8 >= 0;      y3_17 >= 0;      y17_18 >= 0;
y18_19 >= 0;     y19_20 >= 0;     y20_21 >= 0;
y21_22 >= 0;     y22_23 >= 0;     y23_17 >= 0;
y21_6 >= 0;      y15_24 >= 0;     y24_19 >= 0;
y2_24 >= 0;      y7_24 >= 0;

/* Restricoes de integralidade */
int  y0_1   , y1_2.  , y1_10  , y2_3   , y2_24  , y3_4   , y3_17  ,
     y5_6   , y6_7   , y7_24  , y7_8.  , y8_9   , y9_0   , y10_11 ,
     y12_13 , y13_8  , y13_14 , y14_15 , y15_16 , y16_10 , y15_24 ,
     y24_19 , y19_20 , y20_21 , y21_6  , y21_22 , y22_23 , y23_17 ,
     y17_18 , y18_19 , y4_5   , y11_12;


```

3.3.2 *Output* do *lp_solve*

```
Valor da funcao objectivo: 220.00

Valor atual das variaveis:
y0_1   = 4      y1_2   = 2      y1_10  = 1
y2_3   = 1      y2_24  = 0      y3_4   = 0
y3_17  = 0      y4_5   = 0      y5_6   = 0
y6_7   = 4      y7_24  = 0      y7_8   = 3
y8_9   = 4      y9_0   = 4      y10_11 = 2
y11_12 = 2      y12_13 = 2      y13_8  = 0
y13_14 = 1      y14_15 = 1      y15_16 = 0
y16_10 = 0      y15_24 = 0      y24_19 = 2
y19_20 = 4      y20_21 = 4      y21_6  = 3
y21_22 = 0      y22_23 = 0      y23_17 = 0
y17_18 = 1      y18_19 = 1
```

3.3.3 Ficheiro de *Input* do *Relax4*

```
25
32
1 2 3 99
2 3 3 99
2 11 3 99
3 4 4 99
3 25 4 99
4 5 2 99
4 18 2 99
5 6 8 99
6 7 2 99
7 8 4 99
8 9 4 99
8 25 4 99
9 10 2 99
10 1 8 99
11 12 2 99
12 13 3 99
13 14 1 99
14 9 2 99
14 15 2 99
15 16 2 99
16 17 1 99
16 25 2 99
17 11 1 99
18 19 1 99
19 20 2 99
20 21 1 99
21 22 1 99
22 7 3 99
```

```
22 23 1 99
23 24 3 99
24 18 1 99
25 20 3 99
0
-1
-1
-1
0
0
1
-1
1
0
1
0
0
-1
0
-1
0
1
0
1
0
-1
0
0
2
```

3.3.4 Ficheiro de *Output* do *Relax4*

```
************************************************************
 NUMBER OF NODES = 25, NUMBER OF ARCS = 32
 DEFAULT INITIALIZATION USED
 *********************************
 Total algorithm solution time =   0.0155777931 sec.
 OPTIMAL COST =   135.
 NUMBER OF ITERATIONS =  42
 NUMBER OF MULTINODE ITERATIONS =  10
 NUMBER OF MULTINODE ASCENT STEPS =  15
 NUMBER OF REGULAR AUGMENTATIONS =  6
 *********************************
s 135.
f 1 2 4
f 2 3 2
f 2 11 1
f 3 4 1
f 3 25 0
f 4 5 0
f 4 18 0
f 5 6 0
f 6 7 0
f 7 8 4
f 8 9 3
f 8 25 0
f 9 10 4
f 10 1 4
f 11 12 2
f 12 13 2
f 13 14 2
f 14 9 0
f 14 15 1
f 15 16 1
f 16 17 0
f 16 25 0
f 17 11 0
f 18 19 1
f 19 20 1
f 20 21 4
f 21 22 4
f 22 7 3
f 22 23 0
f 23 24 0
f 24 18 0
f 25 20 2
---------------------- end dimacs-format results --------------

```

3.4 Interpretação de resultados

Nesta secção irão ser analisados os resultados obtidos tanto pelo *lp_ solve* como pelo *relax4*. Numa primeira fase, será analisada, separadamente, o valor das funções objectivo de cada uma das soluções e, posteriormente, visto que o resultado das variáveis de decisão é o mesmo, iremos a analisar o resultado obtido, transformando os fluxos dos arcos em caminhos entre vértices de excesso e defeito.

3.4.1 Função Objectivo (*lp_ solve*)

O valor da função objectivo obtido pelo *lp_ solve* foi de 220, o mesmo valor obtido na primeira modulação do trabalho. Este valor é referente ao custo de percorrer a rede apenas uma vez, ou seja, o somatório dos custos parciais da rede, mais o custo dos reposicionamentos realizados, ou seja, o somatório dos custos das arestas reposicionadas vezes o respectivo número de reposicionamentos:

$$\begin{aligned} z^* &= \sum_{(i,j)\in A} c_{i,j} + \sum_{(i,j)\in A} c_{i,j} y_{i,j}, \quad \forall (i,j) \in A \\ &= 85 + 135 \\ &= 220 \end{aligned}$$

3.4.2 Função Objectivo (*relax4*)

O valor da função objectivo obtido pelo *relax4* é ligeiramente diferente pois apenas calcula apenas uma das componentes da função objectivo, *i.e*, o resultado obtido apenas diz respeito ao custo óptimo de reposicionamento não contabilizando a componente que assegura que todos os arcos são visitados.

Com isto, a função objectivo calculada pelo *relax4* é referente à seguinte expressão:

$$\begin{aligned} z^* &= \sum_{(i,j)\in A} c_{i,j} y_{i,j}, \quad \forall (i,j) \in A \\ &= 135 \end{aligned}$$

Para então obtermos o custo efectivo do deslocamento do veículo é necessário somar a este valor a quantidade de 85, referente ao somatório dos custos parciais da rede. Dada esta observação, pode-se então deduzir que o custo total de deslocamento é de $135 + 85 = 220$.

3.4.3 Caminhos de reposicionamento

De seguida, é apresentada a rede com os valores de reposicionamento obtidos por ambos os *softwares*.

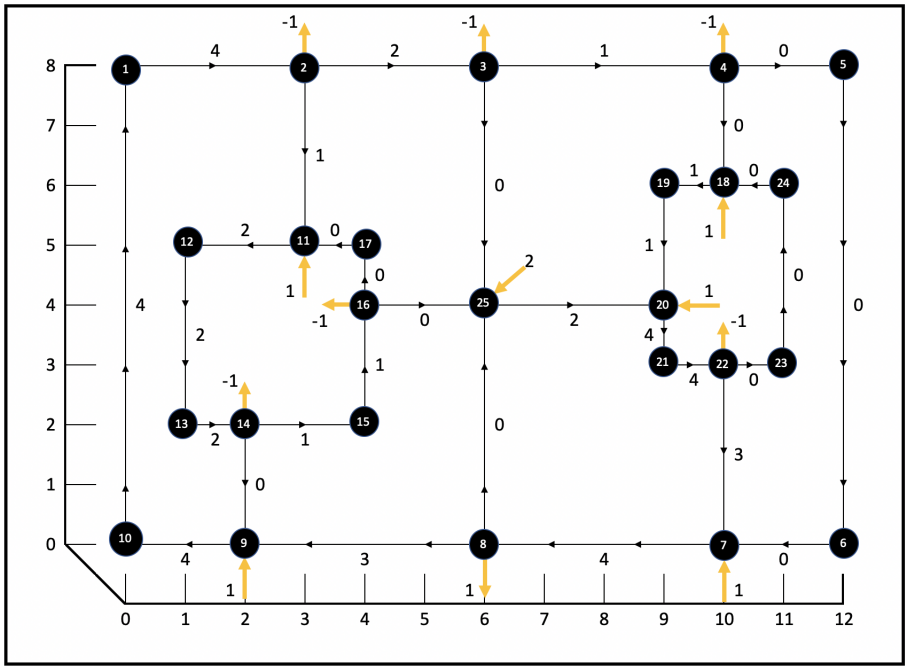

Figura 9: Mapa com fluxo de reposicionamento

Como foi referido anteriormente, os fluxos de uma rede podem ser representados num conjunto de caminhos entre vértices de excesso e defeito (*Teorema da Decomposição de Fluxo*).

O procedimento que vai ser usado para definir os caminhos através dos fluxos das arestas vai ser o seguinte:

1. Fixar um vértice de excesso da rede.
2. A partir do vértice escolhido, efectuar uma pesquisa em profundidade (*Depth-First Search*) de maneira a obter-se o caminho mais longo possível.
3. Quando se passa numa determinada aresta, o seu fluxo é decrementado uma unidade, representando o envio de uma unidade de fluxo entre um vértice de excesso e um de defeito.
4. Garantir que o caminho é balanceado, *i.e*, o somatório das ofertas dos vértices de excesso tem que ser igual ao somatório dos consumos dos vértices de defeito.

5. Aplicar este procedimento enquanto existirem vértices de excesso não visitados e enquanto houverem fluxos positivos.

Com base no que foi referido anteriormente, pode-se então começar a traçar, iterativamente, os caminhos óptimos de reposicionamento. Irão ser escolhidos primeiro os vértices de excesso mais à direita do mapa durante a execução do procedimento. Em cada figura abaixo apresentada, é apresentado o fluxo actual de cada aresta, após o caminho ser definido.

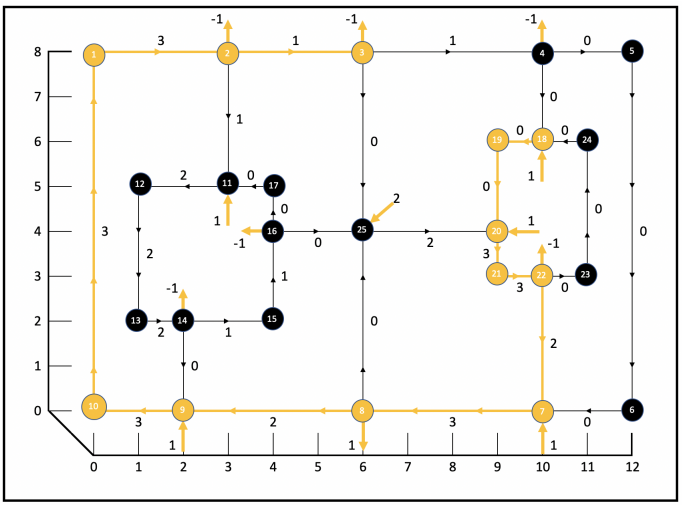

Figura 10: Caminho de reposicionamento 1: custo = 32

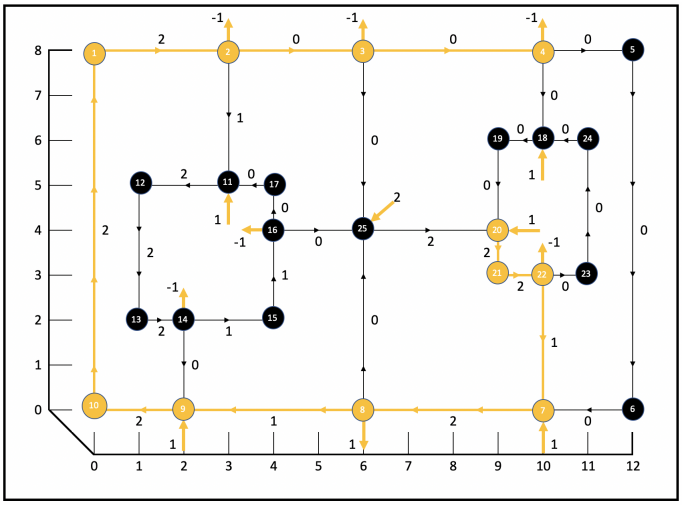

Figura 11: Caminho de reposicionamento 2: custo = 33

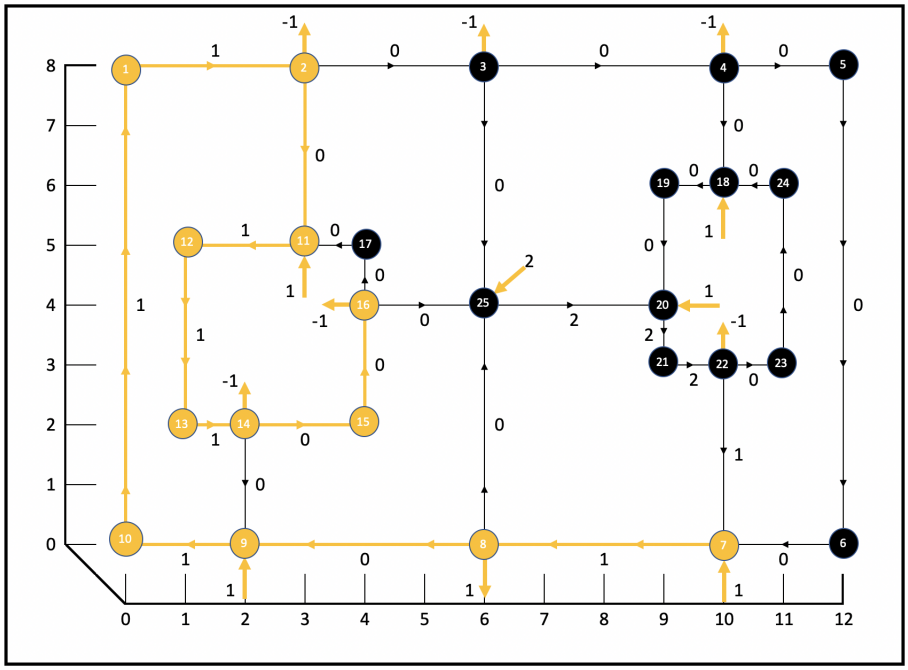

Figura 12: Caminho de reposicionamento 3: custo = 34

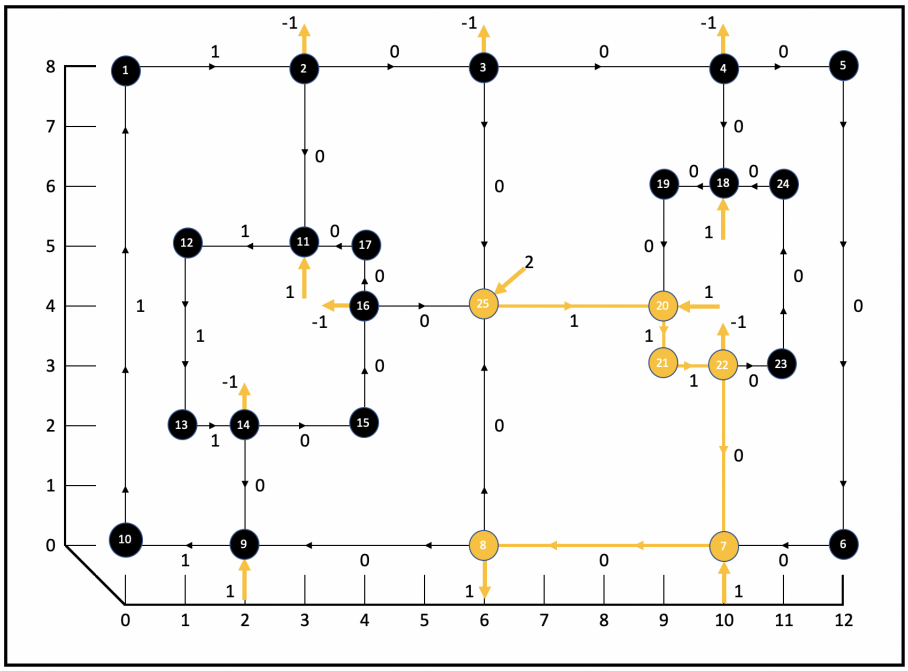

Figura 13: Caminho de reposicionamento 4: custo = 12

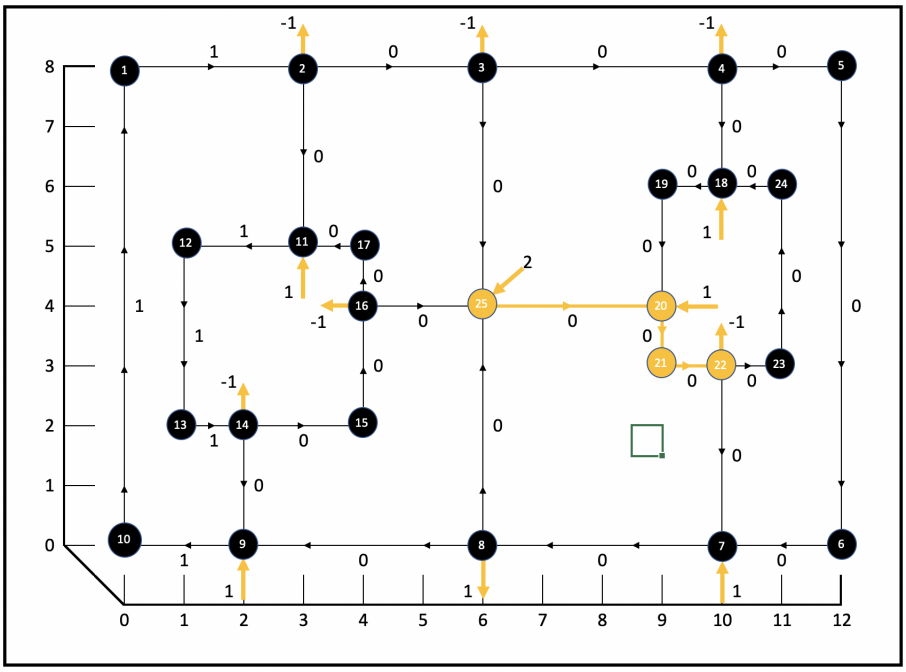

Figura 14: Caminho de reposicionamento 5: custo = 5

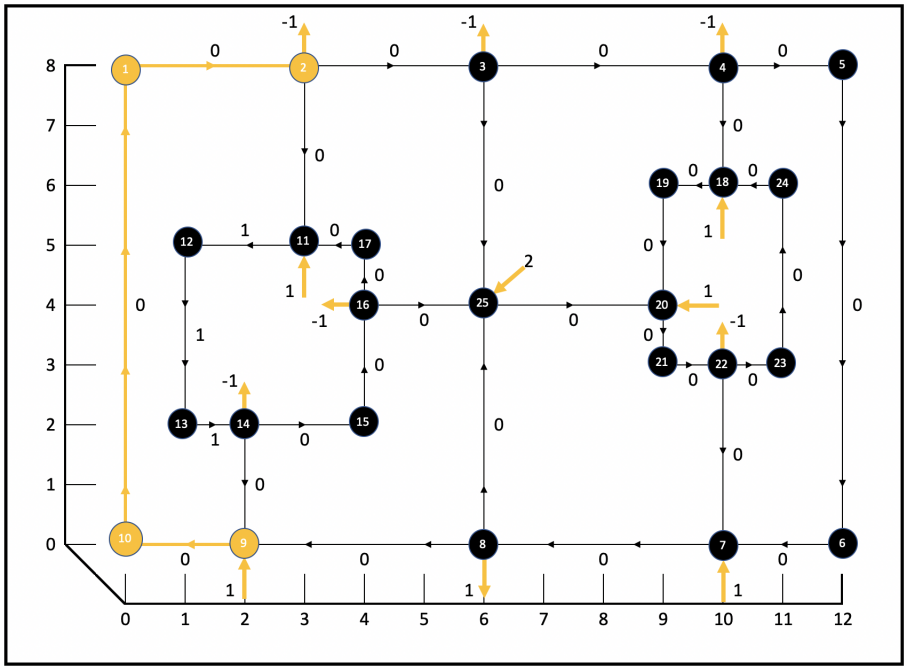

Figura 15: Caminho de reposicionamento 6: custo = 13

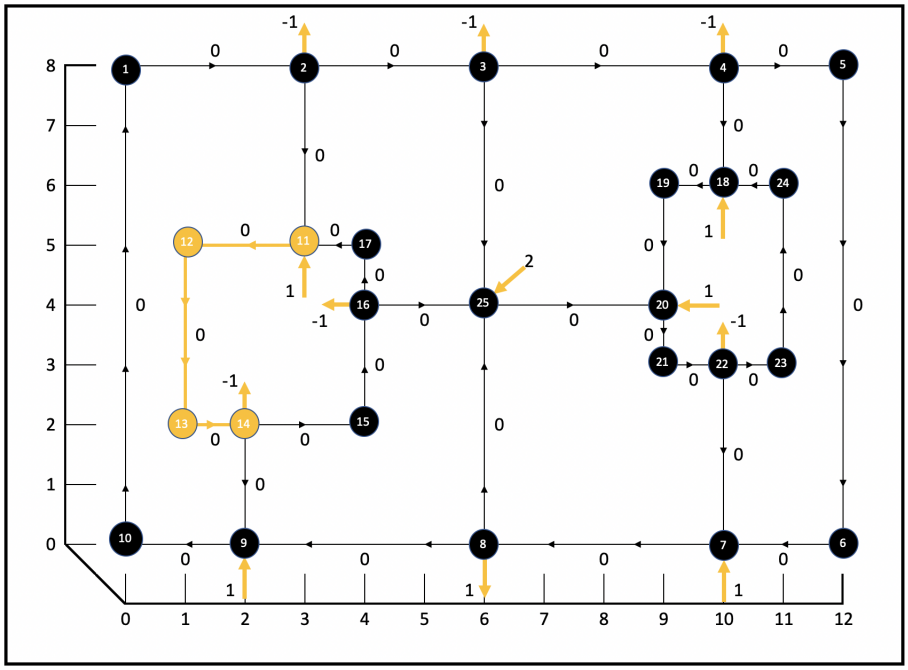

Figura 16: Caminho de reposicionamento 7: custo = 6

Ao fim de todas as iterações obtém-se um conjunto de caminhos óptimos de reposicionamento do veículo na rede e, para além disso, pode-se também observar que todos os fluxos da rede têm valor de zero, significando que todos os reposicionamentos foram feitos.

Sendo $P_i, i \in \{1, ..., 7\}$ um caminho obtido pela solução, c_{Pi} o custo desse caminho e c_R o custo total de reposicionamento, pode-se definir um caminho como a ligação entre os vértices de extremidade, como se pode absorver:

$P_1 : 18 \rightarrow 3$	$c_{P1} : 32$
$P_2 : 20 \rightarrow 4$	$c_{P2} : 33$
$P_3 : 7 \rightarrow 16$	$c_{P3} : 34$
$P_4 : 25 \rightarrow 8$	$c_{P4} : 12$
$P_5 : 25 \rightarrow 22$	$c_{P5} : 5$
$P_6 : 9 \rightarrow 2$	$c_{P6} : 13$
$P_7 : 11 \rightarrow 14$	$c_{P7} : 6$

Para além disso, o custo total de reposicionamento é dado pela soma dos custos de cada um dos caminhos:

$$\begin{aligned} c_R &= \sum_{i=1}^{7} c_{Pi} \\ &= 32 + 33 + 34 + 12 + 5 + 13 + 6 \\ &= 135 \end{aligned}$$

Por fim, podemos concluir que a solução obtida é válida pois obedece tanto às restrições de conservação de fluxo como às restrições de integralidade e não negatividade. Para além disso, todos os caminhos de reposicionamento são válidos pois respeitam o balanceamento entre vértices de excesso e vértices de defeito.

4 Trabalho Prático II (Parte II)

4.1 Definição do problema

Para além das últimas duas formulações, existe ainda outra que revela a estrutura óptima do problema. Esta formulação passa por considerar uma nova variável de decisão, $x_{i,j}$, que representa o caminho (mais curto) entre o vértice de excesso i e o vértice de defeito j. Esta formulação deriva do facto de haverem vértices não balanceados, *i.e*, quando num cruzamento o número de ruas que entram é maior que o número de ruas que saem, é necessário repetir a passagem de pelo menos uma das ruas de saída. O oposto acontece quando o número de ruas que saem de um cruzamento é maior do que o número de ruas que entram, pelo que é necessário voltar a repetir pelo menos uma das ruas que entram.

4.2 Formulação do problema

O objectivo desta formulação é descobrir um conjunto de caminhos de reposicionamento do veículo na rede, sendo que estes caminhos ligam sempre um vértice de excesso e um vértice de defeito. Uma vez que temos dois conjuntos disjuntos de vértices, podemos modular a rede como um grafo bipartido:

$$G = (V_1, V_2, A)$$

onde V_1 é conjunto de vértices de excesso, V_2 é o conjunto de vértices de defeito e A é o conjunto de todas as arestas que ligam os vértices de excesso e defeito.

Uma vez que a representação gráfica do grafo não seria tão legível, decidiu-se usar a representação tabular do mesmo. Pode-se observar na tabela apresentada abaixo, que cada vértice de excesso tem ligação para todos os vértices de defeito. Para além disso, cada célula contém o custo do caminho mais curto entre cada par de vértices. Dada a baixa complexidade da rede, estes valores foram calculados por inspecção da rede, no entanto, seria necessário usar um algoritmo de caminho mais curto para obter estes valores numa rede mais complexa.

	2	3	4	8	14	16	22	
7	21	24	28	4	30	34	13	1
9	13	16	20	32	22	26	25	1
11	21	24	28	24	6	10	17	1
18	29	32	36	12	38	42	5	1
20	26	29	33	9	35	39	2	1
25	29	32	36	12	38	42	5	2
	-1	-1	-1	-1	-1	-1	-1	

Para o problema ter uma solução admissível, é necessário que a rede seja balanceada, *i.e*, a soma dos valores de oferta dos vértices de excesso tem que ser igual à soma dos valores de consumo dos vértices de defeito. Esta propriedade é verificada pois:

$$\begin{aligned} \sum_{i=1}^{6} b_i &= |\sum_{j=1}^{7} b_j|, \quad i \in V1 \wedge j \in V2 \\ 1+1+1+1+1+2 &= |-1-1-1-1-1-1-1-1| \\ 7 &= 7 \end{aligned}$$

Caso isso não acontecesse, ou seja, se a soma dos valores de oferta fossem menores do que a soma dos valores de procura, o problema seria impossível e não seria possível para os *softwares* usados calcular uma solução para o problema.

4.2.1 Variáveis de Decisão

As variáveis de decisão $x_{i,j}$ representam o número de vezes que um caminho entre um vértice de excesso e um vértice de defeito é revisitado, sendo este caminho uma composição de arestas da rede.

4.2.2 Função Objectivo

A função objectivo é dada pela soma dos custos dos caminhos reposicionados. Pode-se então expressá-la como:

$$\min \quad \sum_{(i,j)\in A} c_{i,j}x_{i,j}, \quad \forall (i,j) \in A, i \in V_1, j \in V_2$$

4.2.3 Restrições

As restrições mantém-se as mesmas da última formulação, ou seja, mantém-se as restrições de conservação de fluxo e as restrições de não negatividade. A única diferença é que serão eliminadas as restrições referentes aos vértices balanceados.

$$
\begin{aligned}
& -\sum_{(k,i)\in A} x_{k,i} + \sum_{(i,j)\in A} x_{i,j} = b_i, && \forall i \in V \\
& 0 \leq x_{i,j} \leq 99, && x_{i,j} \in \mathbb{N}
\end{aligned}
$$

4.2.4 Estrutura das soluções

Na última formulação do problema, $y_{i,j}$ representava a atribuição de um fluxo a cada aresta da rede, fluxo esse que representava o reposicionamento do veículo numa aresta. Nesta formulação existe uma ligeira diferença, $x_{i,j}$ representa uma selecção de caminhos de reposicionamento entre vértices de excesso e defeito, ou seja, neste caso não é necessário compor caminhos através dos fluxos pois as soluções do problema já dá esta informação.

4.2.5 Modelo final

Por fim, pode-se representar o modelo final, semelhante ao que já se tinha visto. A função objectivo diz respeito ao custo dos caminhos de reposicionamento e as restrições mantém-se as mesmas, referentes à conservação de fluxo e não-negatividade das soluções.

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{i,j} x_{i,j} \\
\text{s.a} \quad & -\sum_{(k,i)\in A} x_{k,i} + \sum_{(i,j)\in A} x_{i,j} = b_i, && \forall i \in V \\
& 0 \leq x_{i,j} \leq 99, && x_{i,j} \in \mathbb{N}
\end{aligned}
$$

4.3 Ficheiro de *Input*

Para que o *script* de *input* do *relax4* seja válido, foi necessário alterar a numeração dos vértices. Dado isso, apresenta-se de seguida a relação entre a numeração dos vértices no *script* e os vértices do mapa.

Vértices Mapa	Vértices do script
2	7
3	8
4	9
7	1
8	10
9	2
11	3
14	11
16	12
18	4
20	5
22	13
25	6

```
13
42
1 7 21 99
1 8 24 99
1 9 28 99
1 10 4 99
1 11 30 99
1 12 34 99
1 13 13 99
2 7 13 99
2 8 16 99
2 9 20 99
2 10 32 99
2 11 22 99
2 12 26 99
2 13 25 99
3 7 21 99
3 8 24 99
3 9 28 99
3 10 24 99
3 11 6 99
3 12 10 99
```

```
3 13 17 99
4 7 29 99
4 8 32 99
4 9 36 99
4 10 12 99
4 11 38 99
4 12 42 99
4 13 5 99
5 7 26 99
5 8 29 99
5 9 33 99
5 10 9 99
5 11 35 99
5 12 39 99
5 13 2 99
6 7 29 99
6 8 32 99
6 9 36 99
6 10 12 99
6 11 38 99
6 12 42 99
6 13 5 99
1
1
1
1
1
2
-1
-1
-1
-1
-1
-1
-1
```

4.4 Ficheiro de *Output*

```
************************************************************
 NUMBER OF NODES = 13, NUMBER OF ARCS = 42
 UNKNOWN OR UNSPECIFIED INITIALIZATION MODE;USING DEFAULT INITIALIZATION
 *********************************
 Total algorithm solution time =   0.00382518768 sec.
 OPTIMAL COST =   135.
 NUMBER OF ITERATIONS =  18
 NUMBER OF MULTINODE ITERATIONS =  2
 NUMBER OF MULTINODE ASCENT STEPS =  0
```

```
 NUMBER OF REGULAR AUGMENTATIONS =  3
 ***********************************

-------------------- begin dimacs-format results --------------------
s 135.
f 1 7 0
f 1 8 0
f 1 9 0
f 1 10 0
f 1 11 0
f 1 12 1
f 1 13 0
f 2 7 1
f 2 8 0
f 2 9 0
f 2 10 0
f 2 11 0
f 2 12 0
f 2 13 0
f 3 7 0
f 3 8 0
f 3 9 0
f 3 10 0
f 3 11 1
f 3 12 0
f 3 13 0
f 4 7 0
f 4 8 1
f 4 9 0
f 4 10 0
f 4 11 0
f 4 12 0
f 4 13 0
f 5 7 0
f 5 8 0
f 5 9 1
f 5 10 0
f 5 11 0
f 5 12 0
f 5 13 0
f 6 7 0
f 6 8 0
f 6 9 0
f 6 10 1
f 6 11 0
f 6 12 0
f 6 13 1
---------------------- end dimacs-format results ----------------------
```

4.5 Interpretação de resultados

Pode-se observar que o resultado obtido com esta modelação é exactamente igual ao da modulação anterior, portanto, os caminhos óptimos de reposicionamento que seriam gerados com este modelo são exactamente iguais aos que estão representados nas figuras 10, 11, 12, 13, 14, 15, 16. O valor da função objectivo continua a ser 135, que representa o custo dos caminhos de reposicionamento sendo que seria necessário somar o custo total da rede para obtermos o custo total do percurso do veículo, ou seja, $z^* = 135 + 85 = 220$.

Os resultados obtidos são válidos pois respeitam todas as restrições impostas ao problema.

Esta formulação traz uma vantagem em relação à última formulação, o resultado obtido já indica os caminhos óptimos de reposicionamento, enquanto na outra solução tinha-se que derivar os caminhos de reposicionamento através do fluxo das arestas.

Outro aspecto interessante desta formulação é que, é possível através dos caminhos de reposicionamento gerados, calcular os circuitos obtidos no primeiro trabalho prático. Para isto, bastava ligar diferentes caminhos de reposicionamento, onde os pontos de ligação desses mesmos caminhos seriam as arestas com o fluxo de zero, *i.e*, as arestas que apenas é necessário que o veículo as visite uma única vez. Podia-se tirar partido do grafo bipartido para a definição dos circuitos, fazendo com que um vértice de defeito se ligasse a um vértice de excesso, onde esta ligação era referente a uma aresta de fluxo igual a zero.

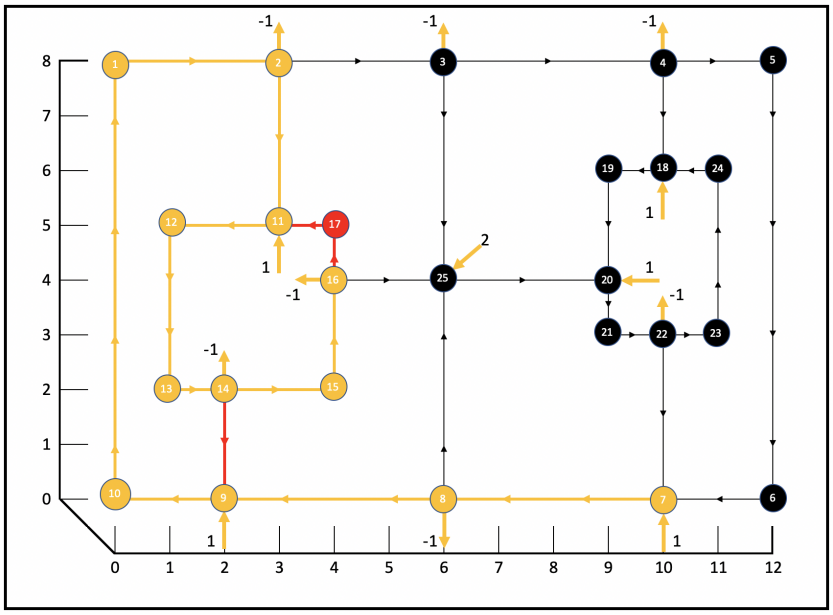

Figura 17: Sub-Mapa da cidade

Para melhor exemplificar o facto de um circuito poder ser obtido através de uma composição de caminhos mais curtos entre vértices de excesso e defeito, ilustra-se de seguida uma possível composição. Na figura 17, pode-se observar uma composição de três caminhos diferentes, sendo estes caminhos os representados nas figuras 12, 15 e 16, referentes, respectivamente, aos caminhos P_3 que liga o vértice 7 ao 16, P_6 que liga o vértice 9 a 2 e P_7 que liga os vértices 11 e 14. Se ligarmos os caminhos P_3 e P_7 através da aresta que liga os vértices 16 e 11 e os caminhos P_7 e P_6 através da aresta que liga os vértices 14 e 9 obtém-se um circuito possível para o veículo.

5 Conclusão

Na primeira parte do trabalho, o objectivo era definir um conjunto de caminhos óptimos para que um veículo de recolha de lixo consiga recolher o lixo de todas as ruas de uma cidade de maneira a minimizar os custos da operação. Foi definido um modelo matemático de programação linear para modelar o problema e, após a definição do modelo, foi feito um *script* para o *software lp_solve* para resolver o problema. Com os resultados obtidos foi possível então traçar os diferentes circuitos que o veículo teria que percorrer.

Na segunda parte do trabalho prático, o problema era ligeiramente diferente. O objectivo do trabalho não era definir o conjunto de circuitos a serem percorridos pelo veículo mas sim o conjunto de caminhos que teriam que ser repetidos pelo mesmo de maneira a garantir que todas as arestas eram visitadas. Na primeira parte foi calculado o fluxo de cada aresta que representava o número de reposicionamentos do veículo na mesma. Através desse fluxo foi possível traçar caminhos de reposicionamento do veículo (*Teorema da Decomposição de Fluxos*). Na segunda parte, o objectivo já não era descobrir os fluxos das arestas mas sim os caminhos de reposicionamento em si. Para isso, foi possível modelar a rede como um grafo bipartido pois havia dois conjuntos disjuntos de vértices, vértices de excesso e defeito. Para ambos os problemas da segunda parte usaram-se os *softwares lp_solve* e *relax4*. Independentemente da modelação dos problemas serem diferentes, os resultados obtidos foram os mesmos.

Referências

[1] A. J. M. Guimarães Rodrigues, *Investigação operacional, Vol. 1* Universidade do Minho, 1993

[2] Jon Kleinberg Éva Tardos, *Algorithm Design, 1st ed.,* Cornell University