UNIVERSIDADE D
COIMBRA

Inês Margarida Neto Ferrão

HOMOMORFISMOS DE GRAFOS POR CAMINHOS E O PROBLEMA DA IMERSÃO DE REDES

Dissertação no âmbito do Mestrado em Matemática, Ramo Estatística, Otimização e Matemática Financeira orientada pelo Professor Doutor João Eduardo da Silveira Gouveia e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia.

Setembro de 2019

Homomorfismos de grafos por caminhos e o problema da imersão de redes

Inês Margarida Neto Ferrão

UNIVERSIDADE D
COIMBRA

Mestrado em Matemática

Master in Mathematics

Dissertação de Mestrado | MSc Dissertation

Setembro 2019

Agradecimentos

Gostaria de agradecer a todos aqueles que, de alguma forma, permitiram que este trabalho se concretizasse.

Em particular, quero agradecer ao Professor Doutor João Eduardo da Silveira Gouveia pela orientação prestada, pelo seu incentivo e pela sua disponibilidade.

Pela ajuda e partilha de conhecimentos informáticos, quero agradecer à Professora Doutora Marília Pascoal Curado e ao Professor Doutor Tiago José dos Santos Martins da Cruz. E ainda, agradecer ao David Henrique de Souza Lima, pesquisador do CISUC e estudante de doutoramento do DEI, por me ter facultado uma base de dados e um código em R, desenvolvido por ele, que me permitiu testar o problema desenvolvido num caso realista.

Agradeço à minha família pelo apoio, força e carinho que sempre me deram ao longo da minha vida e, em particular, durante este último ano.

Por fim, mas não menos importante, quero agradecer a todos os meus amigos e colegas pela amizade e estímulo que sempre me transmitiram.

A todos o meu sincero e profundo Muito Obrigada!

O trabalho aqui desenvolvido foi parcialmente realizado no âmbito do projeto MobiWise: From mobile sensing to mobility advising (P2020 SAICTPAC / 0011/2015), co-financiado pelo COMPETE 2020, Portugal 2020 - Programa Operacional de Competitividade e Internacionalização (POCI), do ERDF (Fundo Europeu de Desenvolvimento Regional) da União Europeia e da Fundação portuguesa para a Ciência e Tecnologia (FCT).

Resumo

Neste trabalho começamos por abordar o problema clássico do homomorfismo de grafos e em seguida, uma sua variação, o problema do homomorfismo por caminhos. Formulamos este último problema como programa linear inteiro e apresentamos alguns testes numéricos que realizamos para estudar o seu comportamento e performance do IBM ILOG CPLEX Otimization Studio na sua resolução.

Seguidamente abordamos o problema da imersão de redes como sendo uma especialização do problema do homomorfismo por caminhos e apresentamos mais alguns testes numéricos.

Por fim, aplicamos o modelo desenvolvido ao problema da localização de processadores na periferia de uma rede móvel, tendo em conta os padrões de mobilidade dos utilizadores. Realizamos novos testes numéricos e ainda mostramos uma aplicação do modelo desenvolvido a um caso realista.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este trabalho foi parcialmente realizado no âmbito do projeto MobiWise. O MobiWise visa a construção de uma plataforma 5G com o intuito de melhorar a mobilidade do utilizador através de uma implementação real da internet das coisas numa cidade inteligente. Para esse efeito, neste trabalho estudamos em concreto o problema da imersão de redes (virtual network embedding problem).

Para estudarmos este problema, vamos começar por introduzir o problema clássico do homomorfismo de grafos para que depois possamos compreender, uma sua variação, o problema do homomorfismo de grafos por caminhos. Abordando estes dois problemas, estamos em condições de apresentar o problema da imersão de rede virtual e também uma sua especialização, o problema da otimização da colocação de data centers, numa rede móvel, tendo em conta os padrões de mobilidade dos utilizadores.

Neste primeiro capítulo, vamos começar por abordar alguns conceitos fundamentais com o intuito de recordar algumas noções que serão necessários para a compreensão do trabalho desenvolvido. Serão também introduzidas notações que consideramos necessárias para que haja um bom entendimento do trabalho. No segundo capítulo, estudamos o problema do homomorfismo de grafos e uma sua variação, apresentando uma formulação do problema como programa linear inteiro. No terceiro capítulo, realizamos testes numéricos com o intuito de aferir o problema formulado no capítulo anterior. Para isso, temos de gerar instâncias do problema que são inspiradas nas aplicações da imersão de rede virtual. No penúltimo capítulo, apresentamos o problema da imersão de redes como sendo uma especialização do problema do homomorfismo de grafos por caminhos. Apresentamos mais alguns testes numéricos. Por fim, no quinto e último capítulo, aplicamos o modelo desenvolvido ao problema de otimização da colocação de data centers numa rede móvel, tendo em conta os padrões de mobilidade dos utilizadores. Também aqui, iremos apresentar mais alguns testes numéricos e ainda aplicamos o mesmo a um caso realista.

1.1 Conceitos fundamentais

Nesta secção vamos abordar algumas noções básicas relacionadas com grafos e ainda apresentamos duas formas usuais de representar os mesmos. Começamos então por recordar a noção de grafo.

Um *grafo* é um par $G = (V, E)$ formado por um conjunto finito $V = V(G) = \{v_1, \ldots, v_n\}$ de vértices ou nós e um conjunto $E = E(G) = \{e_1, \ldots, e_m\}$ de pares não ordenados de vértices, denominados

arestas. Assim, cada aresta do conjunto E é identificada por um par $e = \{v_i, v_j\}$, com $i, j \in \{1, \ldots, n\}$. Ao longo deste trabalho, identificaremos um grafo através da representação gráfica dada no Exemplo 1.

Exemplo 1 *Seja $G = (V, E)$ um grafo com o conjunto de vértices $V = \{1, 2, 3\}$ e o conjunto de arestas $E = \{\{1, 2\}, \{2, 3\}\}$. G pode ser interpretado através da representação dada na Figura 1.1.*

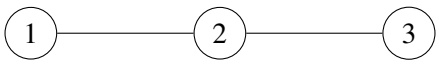


Fig. 1.1 Grafo G com 3 vértices e 2 arestas

Uma aresta diz-se *incidente* a um vértice se esse vértice pertencer à aresta. Dois vértices dizem-se *adjacentes* se existir uma aresta incidente a ambos. Analogamente, duas arestas distintas dizem-se *adjacentes* se tiverem um vértice em comum. Um vértice diz-se *isolado* se não for adjacente a nenhum outro vértice.

Um grafo *orientado* ou *dirigido* define-se da mesma forma que um grafo mas consideram-se as arestas como pares ordenados. Graficamente, a orientação das arestas, representa-se através de uma seta. Um exemplo que ilustra um grafo orientado é o Exemplo 2.

Exemplo 2 *Seja $G = (V, E)$ um grafo orientado com o conjunto de vértices $V = \{1, 2, 3\}$ e o conjunto de arestas $E = \{(1, 2), (2, 3)\}$. G pode ser interpretado através da representação dada na Figura 1.2.*

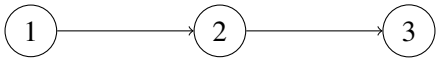


Fig. 1.2 Grafo orientado G com 3 vértices e 2 arestas

Num grafo orientado, o primeiro vértice da aresta é denominado *vértice inicial* e o segundo, *vértice terminal*. A Figura 1.3 ilustra um grafo orientado com uma aresta e dois vértices onde o vértice 1 é o vértice inicial e o 2 o vértice terminal.

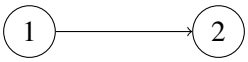


Fig. 1.3 Grafo orientado G com 2 vértices e 1 aresta

Num grafo, o *grau* de um vértice é o número de arestas incidentes a esse vértice e representa-se por $d(v)$. Num grafo orientado, o grau de um vértice, pode ser distinguido entre *grau de saída*, $d^{out}(v)$, e *grau de entrada*, $d^{in}(v)$. O grau de saída conta o número de arestas que saem do vértice e o grau de entrada o número de arestas que entram. Note-se que um vértice isolado tem grau zero. O Exemplo 3 ilustra um grafo orientado com cinco vértices e cinco arestas e contabiliza o grau para alguns dos seus vértices.

Exemplo 3 *Seja $G = (V, E)$ um grafo orientado com o conjunto de vértices $V = \{1, 2, 3, 4, 5\}$ e o conjunto de arestas $E = \{(1, 2), (1, 4), (2, 5), (3, 4), (3, 5)\}$. G pode ser interpretado através da representação dada na Figura 1.4.*

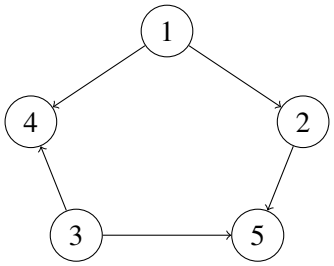


Fig. 1.4 Grafo orientado G com 5 vértices e 5 arestas

Tem-se que $d(2) = 2$, $d^{out}(2) = 1$, $d^{in}(2) = 1$ *e que* $d(4) = 2$, $d^{out}(4) = 0$, $d^{in}(4) = 2$.

Para explicarmos o problema do homomorfismo por caminhos, é necessário introduzirmos o conceito de caminho. Um *caminho* num grafo $G = (V,E)$ é uma sequência de vértices $(v_1, v_2, \ldots, v_t)$ tal que $\{v_i, v_{i+1}\} \in E$ para $i = 1, \ldots, t-1$. Se $v_1 = v_t$, diz-se que é um circuito. Um *grafo cíclico* é um caminho fechado sem repetir vértices, isto é, é um caminho onde $v_1 = v_t$ e $v_1, v_2, \ldots, v_{t-1}$ são distintos dois a dois. Uma *cadeia*, num grafo orientado, é uma sequência $(v_1, e_1, v_2, e_2, \ldots, v_{t-1}, v_t)$ com $v_i \in V, e_i \in E$. A aresta $e_i = (v_i, v_{i+1})$ está orientada no *sentido direto* e a aresta $e_i = (v_{i+1}, v_i)$ no *sentido inverso*. Um *caminho orientado* é uma cadeia onde todas as arestas estão orientadas no sentido direto. Se $v_1 = v_t$, diz-se que é um circuito orientado.

Exemplo 4 *Seja* $G = (V,E)$ *um grafo orientado com o conjunto de vértices* $V = \{1,2,3,4\}$ *e o conjunto de arestas* $E = \{(1,2),(1,4),(2,3),(3,1),(3,4)\}$. *G pode ser interpretado através da representação dada na Figura 1.5.*

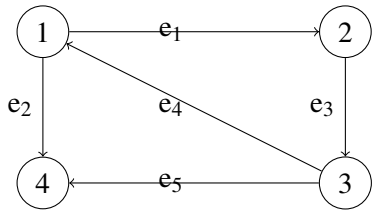


Fig. 1.5 Grafo G com 4 vértices e 5 arestas

Na Figura 1.5, a sequência de vértices $(2, e_3, 3, e_4, 1, e_2, 4)$ *é um caminho orientado e a sequência de vértices* $(1, e_1, 2, e_3, 3, e_4, 1)$ *é um circuito orientado. Quando as arestas são únicas entre dois vértices podemos omitir da sequência a referência às arestas. Assim, as sequências anteriores podem ser substitutas pelas sequências* $(2,3,1,4)$ *e* $(1,2,3,1)$, *respetivamente.*

Um resultado que será útil para nós é uma generalização da versão orientada da caracterização de grafos Eulerianos.

Lema 1 *Se num grafo orientado os graus de saída forem iguais aos graus de entrada para todos os vértices, então o grafo é a união de circuitos orientados disjuntos.*

Um corolário simples deste lema é o seguinte.

Corolário 1 *Se num grafo orientado todos os vértices tiverem graus de saída iguais aos graus de entrada, exceto um que tem* $d^{in}_{(v)} = d^{out}_{(v)} + 1$ *e outro com* $d^{out}_{(u)} = d^{in}_{(u)} + 1$*, então o grafo orientado é a união disjunta de um caminho orientado entre u e v e, possivelmente, circuitos orientados adicionais.*

Duas maneiras úteis de representarmos um grafo são as matrizes de adjacência e incidência. A primeira guarda informação sobre a relação de adjacência entre os nós do grafo e a segunda sobre como os vértices se relacionam com cada aresta, isto é, informações sobre a incidência de uma aresta num vértice.

Considere-se o grafo $G = (V,E)$, com o conjunto de vértices $V = \{v_1, \ldots, v_n\}$. Define-se *matriz de adjacência* como sendo uma matriz $A = [a_{ij}]_{v_i, v_j \in V}$ onde o elemento a_{ij} toma o valor 1 se a aresta $\{v_i, v_j\}$ pertence ao grafo e 0 caso contrário. A matriz anterior tem a particularidade de ser simétrica. Contudo, se estivermos a tratar de um grafo orientado, a sua matriz de adjacência deixa de ser simétrica.

Considere-se o grafo $G = (V,E)$, com o conjunto de vértices $V = \{v_1, \ldots, v_n\}$ e o conjunto de arestas $E = \{e_1, \ldots, e_m\}$. Define-se *matriz de incidência* como sendo uma matriz $B = [b_{ij}]_{v_i \in V, e_j \in E}$ onde o elemento b_{ij} toma o valor 1 se a aresta e_j é incidente ao vértice v_i e 0 caso contrário.

Exemplo 5 *Considere-se o grafo ilustrado na Figura 1.1 do Exemplo 1. A matriz de adjacência é dada por*

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

A matriz de incidência é dada por

$$B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Por vezes torna-se útil associar pesos aos grafos, isto é, atribuir valores numéricos às arestas e/ou aos vértices do grafo e, neste caso, denominamo-los por *grafos ponderados* ou *grafos com pesos*. A Figura 1.6 é uma representação gráfica de um grafo com arestas ponderadas sendo que a aresta $\{1,2\}$ tem peso 20 e a aresta $\{2,3\}$ tem peso 10.

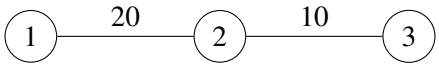


Fig. 1.6 Grafo ponderado com 3 vértices e 2 arestas

Mais sobre conceitos fundamentais de grafos pode ser encontrado em [12].

Capítulo 2

Problema do homomorfismo de grafos

Neste capítulo vamos apresentar o problema clássico do homomorfismo de grafos e ainda uma sua extensão. Vamos também apresentar uma formulação do problema como programa linear inteiro.

2.1 Problema clássico do homomorfismo de grafos

Um homomorfismo de grafos é uma função entre os vértices de dois grafos que respeita a sua estrutura, isto é, preserva a adjacência dos vértices.

Dados os grafos $G = (V(G), E(G))$ e $H = (V(H), E(H))$, um *homomorfismo de G para H* é uma função $f : V(G) \to V(H)$ tal que $\{x,y\} \in E(G) \Longrightarrow \{f(x), f(y)\} \in E(H)$. Se existir um homomorfismo de G para H escreve-se $G \to H$. Neste caso, também se pode dizer que G é homomorfo a H. Caso não exista um homomorfismo de G para H escreve-se $G \not\to H$. Se $G \to H$ e $H \to G$ então G e H dizem-se *homomorficamente equivalentes*.

Exemplo 6 *Considerem-se os grafos G e H, representados na Figura 2.1, que ilustram um pentágono e um triângulo, respetivamente. Note-se que tanto G como H são grafos cíclicos. Pretendemos encontrar um homomorfismo de G para H. De facto, a Figura 2.1 ilustra um homomorfismo de G para H, dado por,* $f(1) = A$, $f(2) = B$, $f(3) = C$, $f(4) = A$, $f(5) = C$.

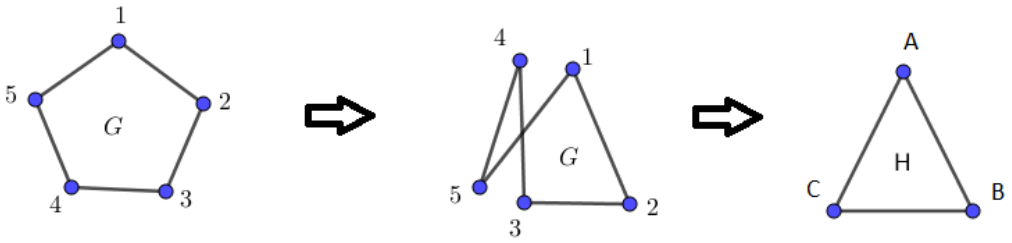


Fig. 2.1 Homomorfismo de G para H, $G \to H$

Um problema clássico da teoria de grafos é o estudo da existência de homomorfismo de grafos. Mais concretamente, estamos interessados no seguinte problema de decisão.

Problema 1 *Dados os grafos $G = (V(G), E(G))$ e $H = (V(H), E(H))$, o problema clássico do homomorfismo de grafos consiste em decidir se existe um homomorfismo de G para H.*

Este problema é notoriamente difícil como ilustramos de seguida.

Para todo grafo G, uma *n-coloração de G* é uma função f de V(G) em $\{1,2,\ldots,n\}$ tal que $\{u,v\} \in E(G) \Rightarrow f(u) \neq f(v)$, para quaisquer $u,v \in V(G)$. Seja K_n o grafo completo de vértices $\{1,2,\ldots,n\}$. Se f é um homomorfismo de G em K_n e se $\{u,v\} \in E(G)$, então $f(u) \neq f(v)$. Logo, f é uma n-coloração de V(G). Por outro lado, se f é uma n-coloração de G, é fácil verificar que f é um homomorfismo de G em K_n. Assim, $G \longrightarrow K_n$ se e só se G é n-colorável. O grafo do Exemplo 6 é 3-colorável, já que é homomorfo a K_3, tal como podemos observar na Figura 2.2.

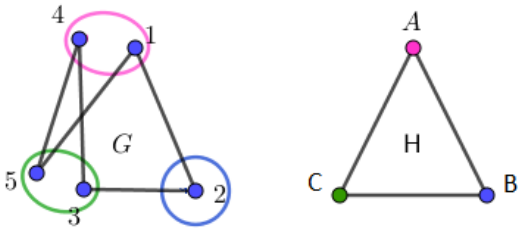


Fig. 2.2 Grafo G 3-colorável

Assim, o problema da coloração de grafos pode reduzir-se ao problema clássico de homomorfismo de grafos. Consequentemente, o primeiro problema referido é pelo menos tão difícil como o segundo. Em particular isto implica que ele é NP-difícil.

Mais sobre o problema clássico do homomorfismo de grafos pode ser encontrado em [3] e [14].

2.2 Problema do homomorfismo por caminhos

Nesta secção, vamos definir o problema do homomorfismos por caminhos para grafos orientados. No entanto, podíamos definir o mesmo para grafos não orientados. Este problema, que também pode ser mencionado como o problema do homomorfismo-P de grafos [7], é uma variante do problema clássico do homomorfismo de grafos.

Dado um grafo orientado H, denotamos o conjunto de todos os seus caminhos orientados por $Ca(H)$. Sejam $G = (V,E)$ e $H = (W,F)$ grafos orientados, chamamos um *homomorfismo por caminhos de G para H* ou *homomorfismo-P de G para H* a um par de mapas $\varphi : V \longrightarrow W$ e $\psi : E \longrightarrow Ca(H)$ tal que $\psi(i,j)$ é um caminho orientado de $\varphi(i)$ para $\varphi(j)$, para todo o $(i,j) \in E$.

Exemplo 7 *Considerem-se os grafos G e H como sendo um triângulo e um quadrado orientados, respetivamente. A Figura 2.3 ilustra um homomorfismo por caminhos de G para H, identificado pela coloração. De facto,* $\psi(1,2) = (A,B,C)$ *é um caminho de* $\varphi(1) = A$ *para* $\varphi(2) = C$, $\psi(2,3) = (C,D)$ *é um caminho de* $\varphi(2) = C$ *para* $\varphi(3) = D$ *e* $\psi(3,1) = (D,A)$ *é um caminho de* $\varphi(3) = D$ *para* $\varphi(1) = A$.

Problema 2 *Dados os grafos orientados* $G = (V(G),E(G))$ *e* $H = (W(H),F(H))$, *o problema do homomorfismo por caminhos consiste em decidir se existe um homomorfismo por caminhos de G para H.*

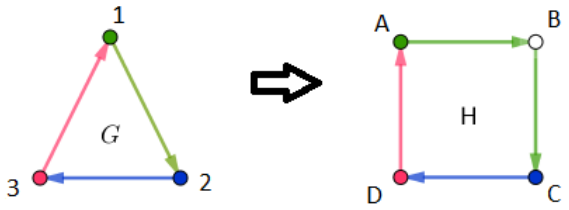


Fig. 2.3 Homomorfismo por caminhos de G para H

Da forma como o problema do homomorfismo por caminhos é definido, é um problema trivial. Basta ter um grafo com um vértice, que qualquer outro grafo pode ser mapeado integralmente para esse vértice, isto porque admitimos caminhos vazios. Tal é ilustrado na Figura 2.4 onde mostramos um mapeamento de um triângulo para um vértice isolado.

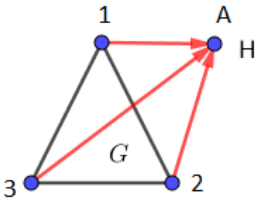


Fig. 2.4 Homomorfismo por caminhos de G para o vértice isolado de H

No entanto, para grafos ponderados, a definição do problema do homomorfismo por caminhos pode ser fortalecida, tornando-a mais interessante.

Considere-se o grafo ponderado $G = (V, E, \theta_G, \omega_G)$ e que, a cada um dos seus vértices $v \in V$, está associado um número real não negativo $\theta_G(v)$, denominado capacidade ou peso do vértice v, e que a cada uma das suas arestas $e \in E$ está também associado um número real não negativo $\omega_G(e)$, denominado capacidade ou peso da aresta e.

Dados os grafos ponderados $G = (V, E, \theta_G, \omega_G)$ e $H = (W, F, \theta_H, \omega_H)$, chamamos um *homomorfismo por caminhos ponderados de G para H* ou *homomorfismo-P ponderado de G para H* a um par de mapas $\varphi : V \longrightarrow W$ e $\psi : E \longrightarrow Ca(H)$ tal que $\psi(i, j)$ é um caminho de $\varphi(i)$ para $\varphi(j)$, para todo o $(i, j) \in E$ e

$$\sum_{v \in \varphi^{-1}(w)} \theta_G(v) \leq \theta_H(w) \quad \forall w \in W, \tag{2.1}$$

$$\sum_{f \in \psi(e)} \omega_G(e) \leq \omega_H(f) \quad \forall f \in F. \tag{2.2}$$

Por outras palavras, a soma dos pesos de todos os vértices que são mapeados para o mesmo vértice w de H não pode exceder a capacidade do mesmo. Analogamente, a soma do peso de todas as arestas que são mapeadas para caminhos contendo uma aresta fixa f de H não pode exceder a capacidade da mesma.

Problema 3 *Dados os grafos ponderados $G = (V, E, \theta_G, \omega_G)$ e $H = (W, F, \theta_H, \omega_H)$, o problema do homomorfismo por caminhos ponderados consiste em decidir se existe um homomorfismo por caminhos ponderados de G para H.*

Exemplo 8 *Considerem-se os grafos G e H do Exemplo 2.1 acrescentando-lhes pesos dados por* $\theta_G(v) = 1\ \forall v \in V$, $\omega_G(e) = 1\ \forall e \in E$, $\theta_H(w) = 2\ \forall w \in W$ *e* $\omega_H(f) = 2\ \forall f \in F$. *Na Figura 2.5 podemos verificar que existe um homomorfismo por caminhos ponderados de G para H.*

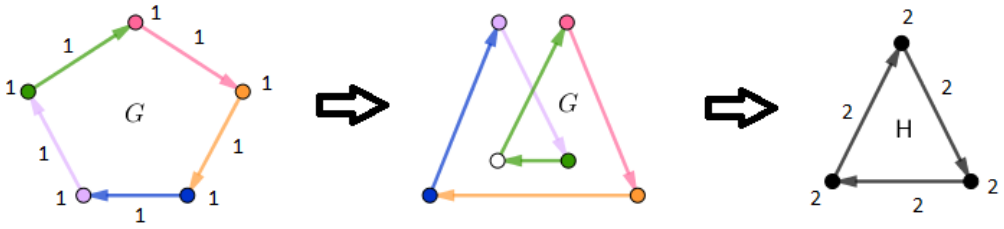


Fig. 2.5 Homomorfismo por caminhos ponderados de G para H

O problema da partição consiste em decidir se um determinado conjunto S de números inteiros positivos pode ser dividido em dois subconjuntos, S_1 e S_2, de maneira que a soma dos números de S_1 seja igual à soma dos números de S_2. Este problema é NP-difícil. Seja σ a soma dos elementos de S. Considere-se G o grafo de $\mid S \mid$ vértices isolados de pesos dados por elementos de S e H o grafo com dois vértices isolados de peso $\frac{\sigma}{2}$ cada um. A existência de um homomorfismo ponderado corresponde à existência de uma partição de S. Logo este problema é NP-difícil mesmo no caso particular sem arestas. Mais sobre o problema da partição pode ler-se em [10]. Na verdade, o problema é NP-Completo como provado em [13] por redução ao problema 3-SAT.

2.3 Formulação como programa linear inteiro

Nesta secção, vamos apresentar uma formulação do problema com o intuito de, posteriormente, tentarmos resolvê-lo, obtendo uma solução para o mesmo. Como já referimos anteriormente, o problema é NP-difícil pelo que será impraticável resolver exatamente o problema como pretendemos para instâncias de tamanho elevado. Contudo, numa primeira abordagem, vamos tentar formulá-lo como um programa linear inteiro para o podermos testar.

2.3.1 Formulação do problema

Vamos então tentar dar uma formulação ao problema inspirado em [1] e [9]. Sejam o grafo de chegada e o grafo de partida denotados por $H = (V_H, E_H, \theta_H, \omega_H)$ e $G = (V_G, E_G, \theta_G, \omega_G)$, respetivamente. O objetivo é mapear o grafo de partida para o grafo de chegada alocando cada vértice/aresta do primeiro em algum vértice/caminho do segundo. Para isso, vamos tentar representar o mapeamento através de variáveis binárias.

Definimos $\triangle_i^k$ como sendo uma variável binária que representa o mapeamento do vértice k do grafo de partida G para o vértice i do grafo de chegada H e $\triangle_{(i,j)}^{(k,l)}$ uma variável binária que representa o mapeamento da aresta (k,l) do grafo G para a aresta (i,j) do grafo H. Pretendemos que verifiquem

- $\triangle_i^k = 1$ se e só se $k \in V_G$ é mapeado para $i \in V_H$;
- $\triangle_{(i,j)}^{(k,l)} = 1$ se e só se $(k,l) \in E_G$ é mapeado para $(i,j) \in E_H$.

Para isto definir um homomorfismo por caminhos, temos de impor as restrições que serão apresentadas de seguida.

Cada vértice de G deverá ser mapeado para um e um só vértice de H. Isto pode traduzir-se através da equação (2.3).

$$\sum_{i \in V_H} \triangle_i^j = 1 \qquad \forall j \in V_G \tag{2.3}$$

De facto, se a soma das variáveis $\triangle_i^j$, para um determinado vértice j de partida, fosse maior do que 1, significaria que existem pelo menos duas variáveis $\triangle_i^j$ e $\triangle_k^j$ com valor 1. Se fosse 0 não existiria nenhuma. Em qualquer um dos casos, não teríamos uma função dos vértices de G para H. Não estaria bem definido.

Pretendemos ainda que as arestas do grafo de partida sejam mapeadas para caminhos entre as imagens dos seus vértices, equação (2.4).

$$\sum_{m \in V_H} \left(\triangle_{(n,m)}^{(k,l)} - \triangle_{(m,n)}^{(k,l)} \right) = \triangle_n^k - \triangle_n^l \qquad \forall (k,l) \in E_G, \forall n \in V_H \tag{2.4}$$

Estas são equações de equilíbrio de fluxo. Fixando (k,l) em G, pretendemos que o conjunto c dos (i,j), tal que $\triangle_{(i,j)}^{(k,l)} = 1$, seja um caminho da imagem de k para a imagem de l. As condições impostas implicam que, para cada vértice de H, entrem tantas arestas de c como saem com a exceção dos vértices que verificam $\triangle_n^k = 1$ ou $\triangle_n^l = 1$, respetivamente a imagem de k e l. Nestes vértices, temos que sai mais uma aresta do que as que entram ou entra uma aresta a mais do que as que saem, respetivamente. Isto implica, pelo Corolário 1, que c é a união disjunta de um caminho da imagem de k para a imagem de l e possivelmente alguns ciclos adicionais. Estes ciclos adicionais, dado que os pesos são não negativos, podem ser removidos sem alterar a existência de uma solução.

Cada vértice i do conjunto dos vértices de chegada, V_H, pode ser imagem de vários vértices do conjunto de partida, V_G. Vamos impor que a capacidade de cada vértice de chegada, $\theta_H(i)$, seja maior ou igual que a soma de todos os pesos dos vértices de G, $\theta_G(j)$, mapeados para esse mesmo vértice de chegada. Isto pode traduzir-se através da desigualdade (2.5). Analogamente, cada aresta (n,m) do conjunto das arestas de chegada, E_H, pode ser imagem de várias arestas do conjunto de partida, E_G. Impomos também que a capacidade, de cada aresta de chegada, $\omega_H(n,m)$, seja maior ou igual que a soma de todos os pesos das arestas de G, $\omega_G(k,l)$, mapeados para essa mesma aresta de chegada. Isto pode traduzir-se através da desigualdade (2.6). Estas desigualdades representam as condições (2.1) e (2.2), respetivamente.

$$\sum_{j \in V_G} \triangle_i^j \theta_G(j) \leq \theta_H(i) \qquad \forall i \in V_H \tag{2.5}$$

$$\sum_{(k,l) \in E_G} \triangle_{(n,m)}^{(k,l)} \omega_G(k,l) \leq \omega_H(n,m) \qquad \forall (n,m) \in E_H \tag{2.6}$$

Temos então a seguinte formulação para o problema do homomorfismo por caminhos.

Proposição 1 *O problema do homomorfismo por caminhos é equivalente ao problema da existência de* $\triangle_i^j \in \{0,1\}$ *para* $j \in V_G, i \in V_H$ *e* $\triangle_{(n,m)}^{(k,l)} \in \{0,1\}$ *para* $\forall (k,l) \in E_G, (n,m) \in E_H$ *tais que*

(1) $\sum_{i \in V_H} \triangle_i^j = 1 \qquad \forall j \in V_G$

(2) $\sum_{m \in V_H} \left(\triangle_{(n,m)}^{(k,l)} - \triangle_{(m,n)}^{(k,l)} \right) = \triangle_n^k - \triangle_n^l \qquad \forall (k,l) \in E_G, \forall n \in V_H$

(3) $\sum_{j \in V_G} \triangle_i^j \theta_G(j) \leq \theta_H(i) \qquad \forall i \in V_H$

(4) $\sum_{(k,l) \in E_G} \triangle_{(n,m)}^{(k,l)} \omega_G(k,l) \leq \omega_H(n,m) \qquad \forall (n,m) \in E_H$

Este é um problema linear inteiro que pode ser resolvido usando as ferramentas existentes para o efeito, como explicaremos mais adiante.

Capítulo 3

Testes numéricos

Neste capítulo, para aferirmos o problema formulado no Capítulo 2, vamos apresentar testes numéricos que realizámos ao mesmo. Para isso, é necessário gerar instâncias tendo em conta as aplicações do problema da imersão de redes.

3.1 Geração do grafo de chegada

Motivados pelas redes de comunicação, decidimos usar como modelo para a geração do grafo de chegada o *modelo de Barabási-Albert*. Este modelo é um algoritmo que vai gerar grafos de forma aleatória. Durante a geração, vão sendo adicionados novos vértices ao grafo, aumentando a dimensão do mesmo, e cada um desses novos vértices será ligado por arestas a um ou mais vértices já existentes. A probabilidade de um vértice i se ligar a outro j, através de uma aresta, é proporcional ao grau do vértice j. Assim, quanto mais conectado for um vértice, maior será a probabilidade de esse vértice receber novas ligações a partir dos vértices que vão sendo adicionados. Escolhemos este modelo para a geração porque é um algoritmo que gera redes sem escala de forma aleatória. As redes sem escala são redes onde a grande maioria dos vértices têm poucas ligações enquanto que a minoria tem um elevado número de ligações.

Em termos de algoritmo, este inicia-se com um grafo arbitrário de n_0 vértices ao qual vão sendo adicionados novos vértices, um de cada vez, e ligados sequencialmente a c dos vértices já existentes, onde c é um parâmetro de construção. A cada passo, a probabilidade, p_i, de um novo vértice se ligar a um outro i já existente é proporcional ao número de ligações que o vértice i já possui. Consequentemente, o grau médio do grafo é dado por $2c$. Mais sobre o modelo de Barabási-Albert pode ser encontrado em [2].

É de referir que o algoritmo que vamos utilizar, foi um pouco alterado em relação a este modelo. Mais concretamente, de cada vez que um vértice está a ser adicionado e ligado a outro já existente no grafo, através de uma aresta, em vez de se adicionar essa aresta não orientada, adicionam-se duas orientadas, uma em cada sentido.

Este algoritmo foi implementado, num programa feito em MATLAB. Os parâmetros de geração são os seguintes: o número de vértices que pretendemos que o grafo de chegada tenha; as capacidades que os vértices e as arestas devem ter; o tamanho da semente, isto é, o valor para n_0; e o valor que pretendemos que o grau médio dos vértices tenha. Este último valor tem de ser sempre menor ou

igual ao valor escolhido para a semente. A partir destes parâmetros, o programa gera uma matriz de adjacência.

O grafo da Figura 3.1 foi obtido através do programa explicado anteriormente e é um exemplo de um grafo de chegada com 30 vértices.

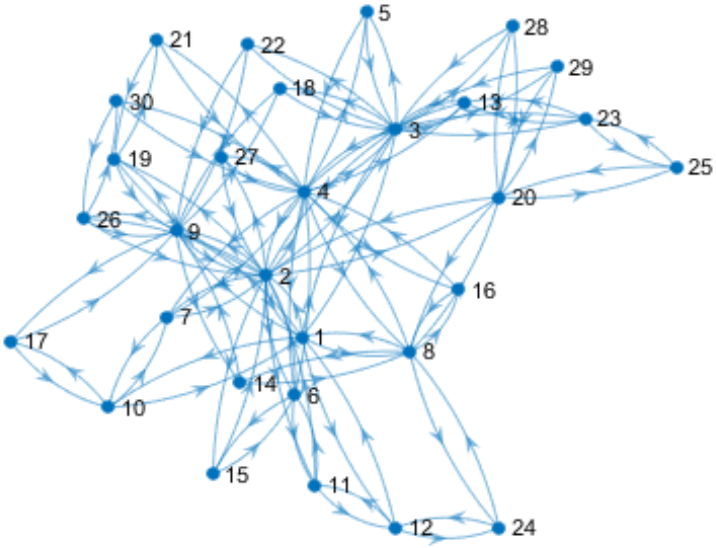


Fig. 3.1 Grafo de chegada

3.2 Geração do grafo orientado de partida

Escolhemos como modelo para o grafo orientado de partida um grafo orientado que fosse composto por cadeias disjuntas. Assim, implementámos um programa em MATLAB para o efeito. Os parâmetros a introduzir são os seguintes: o número de vértices que pretendemos que cada cadeia tenha; o número de repetições da cadeia, isto é, quantas cópias da cadeia queremos que o grafo orientado apresente; e os pesos que cada vértice e aresta devem ter. Tal como no programa da geração do grafo de chegada, é produzida uma matriz de adjacência.

Motivados pelos exemplos clássicos do VNF (Virtual Network Function), definimos no programa implementado, que os vértices inicial e final de cada cadeia têm valor zero como pesos.

O grafo da Figura 3.2 foi obtido através do programa explicado anteriormente e é um exemplo de um grafo orientado de partida composto por 9 cadeias, cada uma com 4 vértices o que perfaz um total de 36 vértices.

3.3 Implementação

Para implementar o problema formulado, recorremos ao programa IBM ILOG CPLEX Optimization Studio e ao MATLAB.

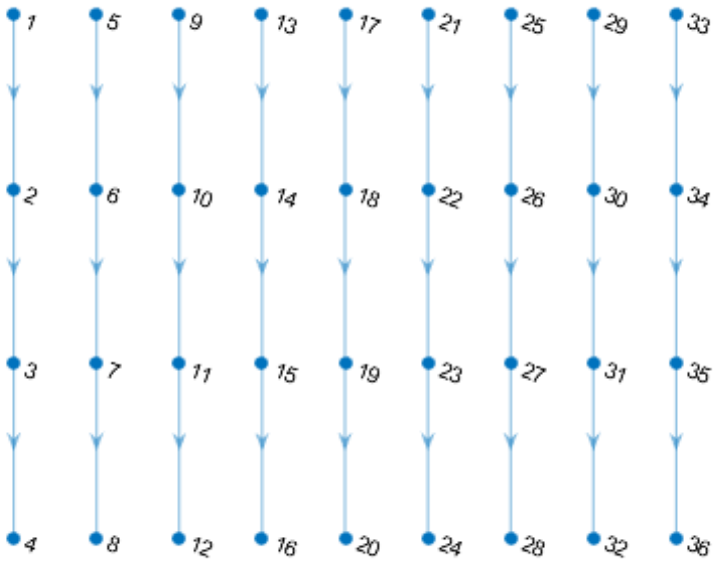


Fig. 3.2 Grafo orientado de partida

Começámos por utilizar o MATLAB para gerar o grafo de chegada e o grafo orientado de partida, como explicámos nas Secções 3.1 e 3.2. Adicionalmente, nestes programas de geração, criámos funções de visualização para conseguirmos ver os grafos gerados. Também implementámos conversões automáticas, dos grafos gerados, em ficheiros .dat para utilizarmos no IBM ILOG CPLEX Optimization Studio.

Posteriormente, utilizámos o IBM ILOG CPLEX Optimization Studio para implementar o problema formulado no Capítulo 2. Este programa recebe toda a informação guardada sobre os grafos gerados anteriormente pelo MATLAB e através da implementação pretendemos obter uma solução para o problema em causa. Aqui, para além de pretendermos obter uma solução, também decidimos minimizar o número de arestas utilizadas no mapeamento.

Correndo então este último programa, obtemos como solução duas matrizes de adjacência, que representam cada uma o conjunto das variáveis $\triangle_i^k$ e $\triangle_{(i,j)}^{(k,l)}$, que nos indicam, através dos seus números binários, para quais vértices e arestas do grafo de chegada foram mapeados os elementos do grafo orientado de partida.

Para tornar a solução visível, criámos um outro programa no MATLAB que lê os ficheiros que guardaram a informação sobre os grafos e a informação da solução que obtivemos. Com estes dados, este programa vai reproduzir o grafo de chegada original com as arestas e os vértices utilizados no mapeamento a vermelho.

A Figura 3.3 foi conseguida através dos programas explicados anteriormente e mostra a solução que obtivemos para o mapeamento do grafo orientado de partida da Figura 3.2 para o grafo de chegada da Figura 3.1.

Tentámos testar o limite deste programa com o intuito de saber até que tamanho máximo poderíamos ir nos grafos a testar. Percebemos que é um programa que permite resolver o problema para grafos de dimensões moderadas.

Na Tabela 3.1 apresentamos alguns tempos de resposta do programa para um exemplo em que o grafo de chegada tinha 500 vértices e 1994 arestas e tinha como capacidade para cada aresta e para cada vértice 3 e 2, respetivamente. Para gerar este grafo considerámos o valor 2 tanto para a semente como para o grau médio dos vértices. Para os grafos orientados de partida assumimos que todas as arestas tinham peso 1 e que todos os vértices tinham peso 1, exceto os vértices iniciais e finais de cada cadeia que tinham peso 0.

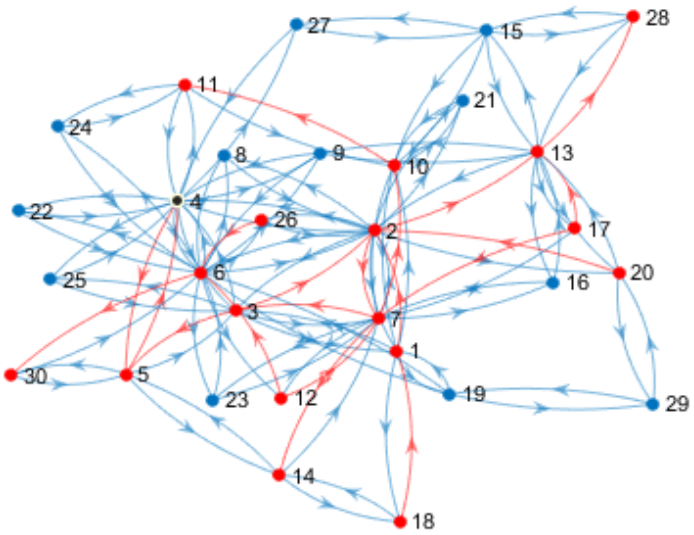


Fig. 3.3 Solução obtida

Nº vértices da cadeia	Nº repetições da cadeia	Nº vértices do grafo orientado de partida	Nº arestas do grafo orientado de partida	Tempo (Segundos)	Solução
5	300	1500	1200	2167.02	erro
4	250	1000	750	510.007	solução
4	300	1200	900	651.001	solução
4	400	1600	1200	962.053	erro
3	500	1500	1000	563.032	solução
3	800	2400	1600	713.04	sem solução
6	300	1800	1500	3006.012	erro

Tabela 3.1 Tabela de testes

Da Tabela 3.1 podemos observar que dos sete casos testados, em três obtivemos solução, noutros três ocorreu um erro e em um outro não obtivemos solução.

No penúltimo teste da Tabela 3.1, onde tentámos mapear um grafo com 2400 vértices e 1994 arestas para outro de 500 vértices e 1600 arestas, não obtivemos solução. Esta falta de solução deveu-se ao facto de não ter sido possível mapear os vértices e as arestas do grafo orientado de partida para vértices e caminhos, respetivamente, do grafo de chegada.

Os três erros obtidos ao fim de um determinado número de segundos, foram motivados pela falta de memória computacional. Uma forma de contornar esta situação, seria utilizarmos um computador com mais capacidade computacional.

É interessante analisarmos o mapeamento de um grafo orientado de partida que seja gerado através do *modelo de Barabási-Albert* para um grafo de chegada gerado sob o mesmo modelo. Esta análise permite-nos perceber de que forma é possível mapear um determinado grafo para outro considerando o mesmo modelo na geração de ambos.

Para isso, continuamos a considerar o modelo descrito na Secção 3.1 para o grafo de chegada. Relativamente ao grafo orientado de partida vamos considerar também o modelo descrito na Secção 3.1, mas aqui não iremos duplicar arestas, isto é, só consideramos arestas num sentido.

A Figura 3.4 foi obtida através do programa explicado anteriormente e é um exemplo de um grafo orientado de partida, com 20 vértices, considerando o *modelo de Barabási-Albert*.

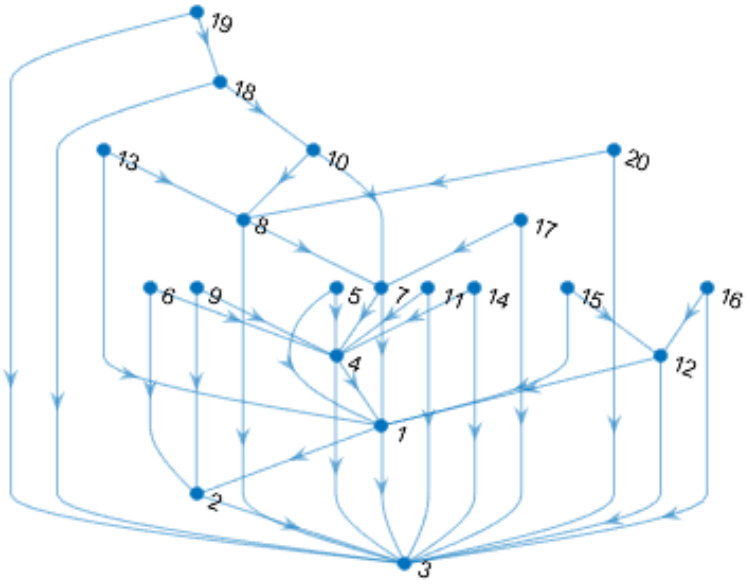


Fig. 3.4 Grafo orientado de partida - Barabási-Albert

Notemos que um grafo gerado considerando o modelo em causa pode ser bastante complexo em termos de ligações entre vértices. Se assim for, mapear um grafo complexo para outro igualmente complexo pode tornar-se uma tarefa difícil ou até mesmo impossível. Isto significa que, caso exista solução do mapeamento, para a obtermos podemos ter de esperar horas. Isto condicionou o tamanho dos grafos de chegada que utilizámos nos testes presentes nas Tabelas 3.2 e 3.3. Decidimos ainda que as capacidades não iriam tomar valores demasiado baixos de modo a não dificultar em demasia o mapeamento.

Nos testes que mostramos de seguida, fixámos o tamanho da semente e o grau médio dos vértices com os valores 3 e 2, respetivamente e para ambos os grafos. Para o grafo de chegada, variámos o número de vértices do mesmo e as suas capacidades para os vértices e para as arestas. Relativamente ao grafo orientado de partida, impusemos que os pesos dos vértices e das arestas fossem ambos 1 em todos os casos testados. O número de vértices deste grafo foi o parâmetro que variámos.

Estes testes foram divididos em duas fases. Numa primeira fase o objetivo foi minimizar o número de arestas a utilizar no mapeamento. Como já referimos, os testes podem tornar-se demorados e se estivermos a minimizar algo, como é o caso da primeira fase mencionada, ainda mais demorados se podem tornar. Assim, decidimos fazer uma segunda fase de testes onde não minimizamos nada. Neste caso, apenas estávamos preocupados em perceber se é possível mapear um grafo para outro.

Podemos observar os resultados dos testes correspondentes à primeira fase na Tabela 3.2 e à segunda fase na Tabela 3.3.

Nº vértices chegada	Capacidade Vértices	Capacidade Arestas	Nº Vértices Partida	Tempo (Segundos)	Solução
20	12	12	50	1825,042	Solução
20	20	20	50	226,007	Solução
20	20	20	100	+ de 7200	Interrompido
10	12	12	50	93,002	Solução
10	20	20	50	25,0.058	Solução
10	20	20	100	+ de 7200	Interrompido

Tabela 3.2 Tabela de testes fase 1 - com minimização

Nº vértices chegada	Capacidade Vértices	Capacidade Arestas	Nº Vértices Partida	Tempo (Segundos)	Solução
50	12	12	100	7,058	Solução
50	12	12	150	3190,004	Solução
50	20	20	150	10,047	Solução
50	20	20	200	1660,029	Solução
10	12	12	110	2,046	Solução
10	12	12	150	2,008	Sem Solução
10	20	20	200	65,031	Solução
10	20	20	250	3,001	Sem Solução

Tabela 3.3 Tabela de testes fase 2 - sem minimização

Tal como seria de esperar, os testes onde se considerou a minimização, Tabela 3.2, apresentam tempos muito superiores relativamente aos da Tabela 3.3, ainda que tenham sido utilizados grafos de menores dimensões para o primeiro caso.

Dois dos casos testados na Tabela 3.2 foram interrompidos, ao fim do tempo indicado na coluna "Tempo", antes mesmo de ele apresentar a solução. A razão pela qual abortámos a solução foi porque não estávamos interessados no tempo exato até se obter a solução. Estávamos apenas interessados em perceber se os tempos diferenciavam muito uns dos outros e duas horas já era um tempo elevado em comparação com os restantes.

Na Tabela 3.3 verificaram-se duas situações onde não obtivemos solução. A falta de solução para estes dois casos já era previsível visto que não podemos querer mapear, por exemplo, 250 vértices, cada um com peso 1, para um grafo que tem apenas 10 vértices, cada um com capacidade 20. Neste caso estamos a tentar mapear um grafo com peso 250 nos vértices para outro grafo que tem apenas 200 de capacidade nos vértices, o que é impossível.

Analisando as duas tabelas em simultâneo, podemos concluir que as capacidades serão sempre uma limitação para a obtenção de solução e até mesmo uma limitação à obtenção da mesma num curto espaço de tempo. É claro que se considerarmos algum tipo de minimização para o problema, isto também tornará o tempo para obtenção de solução mais elevado.

Por fim, podemos afirmar que não é a complexidade de ambos os grafos que põe em causa a obtenção de solução para o problema. Contudo, não se torna prático mapear este tipo de grafos tendo em conta o tempo que se pode demorar a obter uma solução, principalmente se considerarmos grafos de grandes dimensões e ainda considerarmos parâmetros de minimização.

Capítulo 4

Problema da imersão de redes

Neste capítulo vamos começar por abordar o problema da imersão de redes como sendo uma especialização do problema do homomorfismo de grafos por caminhos, apresentado no Capítulo 2. De seguida, ajustamos a formulação apresentada na Subsecção 2.3.1 para o novo problema em estudo. Ainda neste capítulo, apresentamos mais alguns testes numéricos. Por fim, explicamos o que é uma fronteira Pareto-ótima e qual a sua aplicabilidade na obtenção de uma solução para o problema da imersão de redes.

4.1 Problema da imersão de redes

Identificamos o problema da imersão de redes (Virtual Network Embedding Problem) através da sigla VNE. O problema do VNE é tratado como sendo um processo de virtualização de rede que atualmente é uma das tecnologias mais promissoras para o futuro da internet. Até hoje, esta tecnologia tem sido aplicada principalmente no mercado das telecomunicações.

Na arquitetura clássica, quando computadores ou outros equipamentos necessitam de correr programas, ou os correm no próprio aparelho, com capacidade de processamento, ou pedem a um servidor central para os correr por eles e devolver-lhes uma resposta. A ideia da virtualização de rede é partir estes pedidos, que estamos a designar como programas, em pedaços e espalhá-los na rede de forma a que sejam corridos perto do utilizador mas libertando recursos no aparelho.

A estrutura física ou rede física é constituída por computadores ou processadores, isto é, aparelhos com capacidade de processamento de informação. Estes processadores estão ligados entre si através de ligações físicas ou rádio. Vamos abstrair esta estrutura sob a forma de um grafo orientado, que designamos por grafo físico, onde os nós são os computadores ou processadores e as arestas as ligações entre eles.

A esta rede são feitos pedidos que se traduzem por programas que têm de ser corridos. Como já mencionámos, estes programas são partidos em pedaços de forma a poderem ser corridos em diferentes processadores. Estes pedaços têm dependências entre si, isto é, têm de ser corridos por uma certa ordem, ou seja, o output de um pedaço do programa será o input de outro pedaço. Esta informação será guardada sob forma de um grafo orientado, que designamos por grafo virtual, em que os nós são os pedaços de problema a serem corridos e as arestas representam o transporte de informação do output de um para o input de outro.

Tendo em conta que são necessários recursos para processar cada um dos pedaços dos programas, a distribuição dos mesmos pelos diferentes processadores tem de respeitar as capacidade de processamento destes. A capacidade de processamento pode ser de CPU, de memória ou de armazenamento, por exemplo. Também as ligações entre os processadores têm uma largura de banda limitada.

Desta forma, o problema do VNE consiste em determinar onde e como distribuir cada pedaço de cada programa, ou seja, consiste em distribuir os pedaços de programas pelos processadores de maneira que as dependências entre eles sejam preservadas e que a capacidade de processamento e a largura de banda não seja excedida. Isto significa que queremos determinar o local de processamento de cada pedaço de programa e saber por onde passou a informação entre pedaços de programa dependentes. Assim, o problema do VNE consiste então em imergir/mapear a rede virtual na rede de substrato ou física alocando recursos.

Neste problema podemos considerar duas situações distintas, uma estática e outra dinâmica. Na estática, podemos assumir que, em algum momento, é feito um conjunto de pedidos à rede que têm de ser processados mais ou menos em simultâneo. Na dinâmica, são feitos pedidos à rede em diferentes momentos e que têm de ser processados consoante a sua chegada. No nosso estudo, vamos assumir que estamos no caso da situação estática.

Mais sobre o problema do VNE pode ler-se nos artigos [8], [1], [9] e [4].

O problema do VNE é uma especialização do problema do homomorfismo de grafos por caminhos ponderados que, como já referimos, é uma extensão do problema do homomorfismo de grafos por caminhos. De facto, no mapeamento é necessário alocar recursos o que em termos de interpretação é como se se tratassem de pesos atribuídos aos vértices e às arestas.

4.2 Problema da localização de processadores na periferia da rede

Estamos interessados em estudar o problema da localização de processadores na periferia da rede que é uma variação do problema da localização de infraestruturas. No entanto, interpretamo-lo como um caso particular e simplificado do problema de imersão de redes de comunicações móveis porque nos é mais conveniente.

Como cenário de estudo, vamos considerar que a rede física é a infraestrutura de uma operadora de telemóveis. Cada uma das antenas presentes nessa infraestrutura está ligada, através de ligações físicas ou rádio, a um agregador, agregador esse que pode receber ligações de várias antenas. Por sua vez, estes agregadores, estão conectados entre si e a um ou mais gateways, também através de ligações físicas ou rádio. As antenas são o primeiro ponto de contacto entre o utilizador através de, por exemplo, um telemóvel ou um computador, e a rede onde este se vai ligar para, por exemplo, estabelecer uma chamada ou simplesmente ligar-se à internet. Desta forma, são as antenas que recebem os pedidos dos utilizadores que depois são encaminhados para o respetivo agregador. É no agregador que os pedidos são encaminhados para o local onde são corridos. Por sua vez, o gateway é o ponto de comunicação entre a rede local e a rede global que designamos por nuvem, isto é o mesmo que dizermos que o gateway é a porta de entrada na rede, por exemplo, na internet em geral. Na Figura 4.1 podemos observar um esquema simplificado da arquitetura da rede.

Numa situação clássica os pedidos seriam enviados, através do gateway, para serem processados na rede. Isto pode provocar alguma sobrecarga na rede e provocar atrasos na comunicação, prejudicando

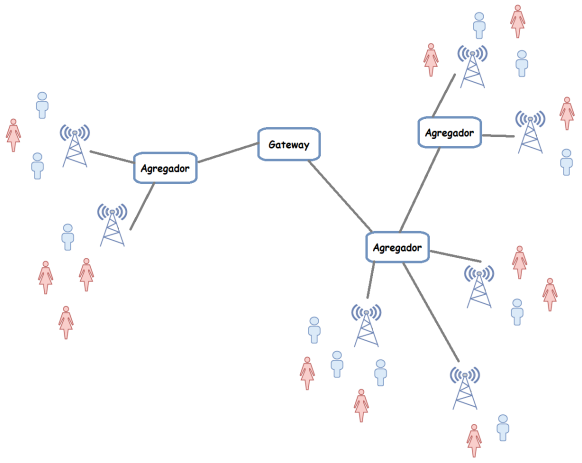


Fig. 4.1 Esquema da arquitetura da rede

a qualidade do serviço. Para evitar este problema, alguns pedidos podem ser corridos em algum processador que esteja entre a antena e o gateway. Esse processador será um data center que tem a capacidade de processar e armazenar grandes quantidades de dados.

No problema da localização de processadores na periferia da rede são os data centers que estamos a localizar. Queremos que eles se localizem perto dos utilizadores para que a informação esteja, geograficamente, perto dos mesmos. Tendo em conta que as antenas são o ponto de contacto entre o utilizador e o data center, então quando dizemos que queremos que estes estejam perto dos utilizadores, eles devem estar perto das antenas, podendo localizar-se inclusivamente nos agregadores da respetiva antena.

É de salientar que uma mesma antena pode ser o ponto de contacto com a rede para diferentes utilizadores. Isto significa que a mesma antena pode receber vários pedidos em simultâneo. Contudo, eles serão condensados e tratados como se se tratasse de um utilizador e por isso de apenas um pedido.

Um exemplo mais concreto deste problema pode ser pensado como uma implementação de um serviço streaming de filmes. Se um operador móvel quiser estabelecer um serviço de clube de vídeo, faz sentido guardar os conteúdos perto dos utilizadores para evitar sobrecarregar a rede com a passagem de dados desde o servidor central até ao utilizador. O que pretenderíamos neste caso, seria estabelecer a melhor localização dos data centers, que seria onde estariam guardados e onde seriam tratados os conteúdos no tal local mais próximo do utilizador.

Para conseguirmos estudar o problema em causa, vamos ter de definir uma arquitetura tanto para a estrutura física como para a virtual através de grafos.

Relativamente ao grafo correspondente à estrutura ou rede física, que designamos por grafo físico, os vértices representam as antenas, os agregadores e o gateway e as arestas as ligações entre eles. Aqui, os vértices vão ter uma capacidade máxima de processamento e e as arestas uma largura de banda limitada.

Designamos o grafo orientado que ilustra a rede virtual por grafo virtual. A Figura 4.2 é a ilustração desse mesmo grafo e explicamos de seguida a arquitetura escolhida para o mesmo. Como já referimos, cada antena recebe vários pedidos por parte dos utilizadores. Supondo que, na rede física, existem n antenas então, neste grafo, tem de haver n cadeias diferentes. Cada cadeia representa

o conjunto de pedidos feitos a cada uma das antenas. O primeiro vértice da cadeia, colorido a azul na Figura 4.2 representa a antena enquanto ponto de contacto entre o pedido feito à mesma pelo utilizador e a rede. O pedido é a agregação de pedidos de muitos utilizadores onde alguns são corridos no processador, o data center, e outros são enviados para serem corridos na rede através do gateway. Assim, os pedidos que podem ser corridos no data center são ilustrados pelo vértice intermédio colorido a preto na mesma figura. Desta forma, a figura tem n vértices pretos, onde cada um corresponde a uma das antenas. Os restantes pedidos, que têm de ser corridos na rede, são ilustrados através do vértice colorido a vermelho na figura em causa. Como já explicámos, o gateway não é o nosso foco de estudo, pelo que assumimos que no nosso problema existe apenas um vértice deste tipo. Por esta razão, não tratamos as cadeias como sendo cadeias disjuntas, mas sim como cadeias que têm o vértice, que representa o gateway, como ponto em comum.

Temos ainda que os vértices e as arestas do grafo virtual têm associados parâmetros que denominamos por pesos. O peso de cada vértice, corresponde à quantidade de recursos de processamento que são necessários para processar os pedidos. O peso de cada aresta, corresponde à quantidade de informação que é necessário passar entre os nós. Estes pesos vão depender do número médio de utilizadores em cada antena porque vamos agregar os pedidos, que chegam a cada uma, de forma a que estes possam ser tratados como valores médios por unidade de tempo.

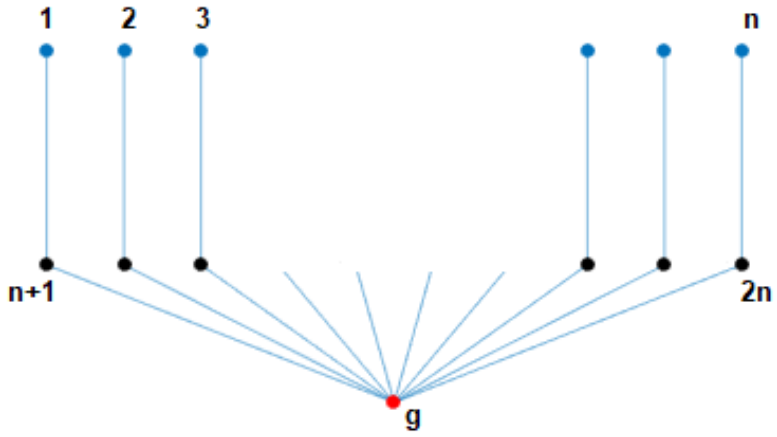


Fig. 4.2 Rede virtual

Vamos então tentar dar uma formulação ao problema da localização de processadores na periferia da rede, tendo por base a formulação da Subsecção 2.3.1.

Na nossa formulação, a rede física e a rede virtual vão ser tratadas como grafos. Assim, denotamos a rede física e a rede virtual por $H = (V_H, E_H, \theta_H, \omega_H)$ e $G = (V_G, E_G, \theta_G, \omega_G)$, respetivamente. Referimos-nos a H como grafo físico e a G como grafo virtual.

Tal como na formulação apresentada na Subsecção 2.3.1, continuamos a considerar as variáveis binárias $\triangle_i^k$ e $\triangle_{(i,j)}^{(k,l)}$ para representar o mapeamento do vértice k do grafo virtual G para o vértice i do grafo físico H e o mapeamento da aresta (k,l) do grafo G para a aresta (i,j) do grafo H, respetivamente. No entanto, para este problema que estamos a abordar, vamos precisar de definir uma nova variável binária $\triangle_i$. Esta variável, através dos seus números binários, permite-nos saber se no vértice i do grafo H está localizado um data center. Assim, pretendemos que se verifique

- $\triangle_i = 1$ se e só se $i \in V_H$ tem um data center lá localizado.

Como já explicámos, no problema que estamos a formular, vamos assumir que existe apenas um gateway. A sua localização está predefinida, já que faz parte da arquitetura da rede.

Na formulação do problema que estamos agora a abordar, vamos considerar novamente algumas das restrições já apresentadas na formulação da Subsecção 2.3.1, bem como acrescentar uma nova restrição e ainda alterar outra.

Vamos então manter as restrições (2.3), (2.4) e (2.6).

A desigualdade (2.5) vai sofrer uma ligeira alteração, embora se mantenha na sua essência. Na nossa abordagem ao problema, definimos a capacidade de um data center como "CapDC". Apenas os vértices onde for localizado um data center, terão capacidade de processamento, pelo que temos a seguinte desigualdade (4.1).

$$\sum_{j \in V_G} \triangle_i^j \theta_G(j) \leq CapDC \times \triangle_i \qquad \forall i \in V_H \tag{4.1}$$

O posicionamento de cada um dos processadores, neste caso dos data centers, é algo mais complexo visto que não queremos colocar muitos, já que acrescentam custos, nem de menos, pois queremos qualidade no serviço. Se quisermos ser nós a definir a quantidade de data centers a utilizar, necessitamos de definir uma constante c para especificar essa mesma quantidade. Para utilizarmos esta constante na nossa formulação, necessitamos de acrescentar mais uma restrição ao problema que se traduz pela equação (4.2). Note-se que, quando estivermos a testar o programa, o parâmetro c terá de ser substituído por um número inteiro positivo inferior ou igual ao número de vértices do grafo físico.

$$\sum_{i \in V_H} \triangle_i = c \tag{4.2}$$

Com esta nova restrição, o nosso problema apenas terá de colocar os c data centers no local ótimo.

No nosso problema alguns vértices do grafo virtual terão de ser mapeados para vértices específicos do grafo físico. Por exemplo, o vértice inicial de cada cadeia corresponde aos pedidos efetuados numa antena específica deverá ser mapeado para esta. Analogamente, existe apenas uma possibilidade para mapear o vértice terminal, que terá de ir para o gateway. Isto pode ser feito através da equação (4.3) que obriga a variável do mapeamento de vértices a tomar o valor 1 para aqueles vértices em específico. Definimos então um conjunto $\mathscr{A}$ constituído por pares de vértices $(i,j) \in V_H \times V_G$ que designamos por âncoras, que codifica o facto do vértice j ter de ser mapeado para o vértice i.

$$\triangle_i^j = 1 \quad \forall (i,j) \in \mathscr{A} \tag{4.3}$$

Temos então a seguinte formulação para o problema da localização de processadores na periferia da rede.

Proposição 2 *O problema da localização de processadores na periferia da rede é equivalente ao problema da existência de* $\triangle_i^j \in \{0,1\}$ *para* $j \in V_G, i \in V_H$, $\triangle_{(n,m)}^{(k,l)} \in \{0,1\}$ *para* $\forall (k,l) \in E_G, (n,m) \in E_H$ *e* $\triangle_i \in \{0,1\}$ *para* $i \in V_H$ *tais que*

$$(1)\ \sum_{i \in V_H} \triangle_i^j = 1 \qquad \forall j \in V_G$$

$$(2)\ \sum_{m \in V_H} \left(\triangle_{(n,m)}^{(k,l)} - \triangle_{(m,n)}^{(k,l)} \right) = \triangle_n^k - \triangle_n^l \qquad \forall (k,l) \in E_G, \forall n \in V_H$$

$$(3^*)\ \sum_{j \in V_G} \triangle_i^j \theta_G(j) \leq CapDC \times \triangle_i \qquad \forall i \in V_H$$

$$(4)\ \sum_{(k,l) \in E_G} \triangle_{(n,m)}^{(k,l)} \omega_G(k,l) \leq \omega_H(n,m) \qquad \forall (n,m) \in E_H$$

$$(5)\ \sum_{i \in V_H} \triangle_i = c$$

$$(6)\ \triangle_i^j = 1 \quad \forall (i,j) \in \mathscr{A}$$

Definimos como objetivo de otimização, para este problema, a minimização da distância que os dados a processar percorrem entre as antenas, os vértices intermédios e o gateway.

Denotamos por p o peso de uma aresta do grafo virtual e por d a distância, isto é, o comprimento que a aresta do grafo físico tem. Assim, pretendemos minimizar f_1 (4.4).

$$f_1 = \sum_{(k,l) \in E_G, (n,m) \in E_H, k,l \in V_G, n,m \in V_H} \triangle_{(n,m)}^{(k,l)} p_{(k,l)} d_{(n,m)} \tag{4.4}$$

No entanto, queremos dar mais importância à minimização da distância entre as antenas e os vértices intermédios. Para isso, temos de reajustar a função definida anteriormente.

Denotamos o conjunto dos vértices virtuais que são antenas, vértices intermédios e gateway por V_A, V_D e V_G, respetivamente. Passamos a querer minimizar f_2 (4.5). Aqui, são as constantes d_1 e d_2 que vão determinar a maior ou menor importância que damos à minimização da distância entre as antenas e os vértices intermédios e os vértices intermédios e o gateway, respetivamente.

$$f_2 = \sum_{(k,l) \in E_G, (n,m) \in E_H} \triangle_{(n,m)}^{(k,l)} \left[d_1 \sum_{k \in V_A, l \in V_D} p_{(k,l)} d_{(n,m)} + d_2 \sum_{l \in V_G, k \in V_D} p_{(k,l)} d_{(n,m)} \right] \tag{4.5}$$

Na implementação do problema podemos não considerar a restrição (4.2), ou seja, não impormos o número de data centers a localizar na rede. Se assim for, temos de voltar a reajustar a função de minimização de forma a passarmos a minimizar também o número de data centers a localizar. Ou seja, à função f_2 (4.5) adicionamos a função f_3 (4.6), ficando com f_4 (4.7) como função de minimização. Aqui, são as constantes w_1 e w_2 que vão determinar a maior ou menor importância que damos à minimização do número de data center a localizar e à minimização da distância entre as antenas e os vértices intermédios e os vértices intermédios e o gateway, respetivamente.

$$f_3 = \sum_{k \in V_H} \triangle_k \tag{4.6}$$

$$f_4 = w_1 \sum_{k \in V_H} \triangle_k + w_2 \sum_{(k,l) \in E_G, (n,m) \in E_H} \triangle_{(n,m)}^{(k,l)} \left[d_1 \sum_{k \in V_A, l \in V_D} p_{(k,l)} d_{(n,m)} + d_2 \sum_{l \in V_G, k \in V_D} p_{(k,l)} d_{(n,m)} \right] \tag{4.7}$$

Assim, quando testarmos o problema formulado, ou consideramos f_2 ou f_4 como função de minimização. É de salientar que se considerarmos f_4 estamos perante um problema biobjetivo, que explicaremos mais detalhadamente na Secção 4.4.

4.3 Testes numéricos

Nesta secção vamos começar por explicar como foram geradas as instâncias e como foi implementado o problema para que, de seguida, possamos apresentar alguns testes numéricos que realizamos para aferir o problema formulado na Secção 4.2.

Para gerar o grafo físico, começámos por desenvolver um algoritmo que gerasse $m \times n$ pontos. Tal como podemos observar na Figura 4.3, os pontos são dispostos de forma que se consigam formar triângulos equiláteros com 1 unidade de lado. Estes pontos representam os vértices do grafo físico, ou seja, as $m \times n$ antenas da rede física. A Figura 4.3 ilustra o posicionamento dos vértices de um grafo com 5×4 pontos. Por uma questão de simplificação, neste primeiro teste, decidimos que a cada antena está associado um agregador. Assim, cada vértice do grafo físico que representa uma antena, também representa o agregador associado à mesma.

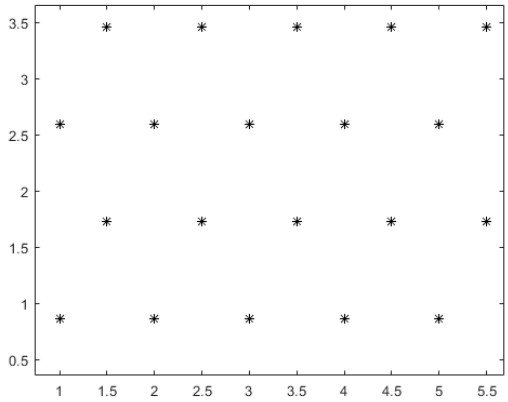


Fig. 4.3 Disposição dos vértices do grafo físico

Como já referimos, vamos assumir que no nosso problema apenas existe um gateway e que este já se encontra predefinido. A sua localização é o vértice mais central do grafo. Este vértice está colorido a vermelho, para que o consigamos distinguir facilmente dos outros, tal como se pode observar na Figura 4.4.

Após a geração dos vértices, é necessário acrescentarmos ligações, isto é, arestas orientadas. No nosso caso consideraremos que cada ligação física pode ser usada em ambos os sentidos, ou seja, que as arestas vêm em pares com os mesmos vértices mas orientações diferentes, pelo que as podemos tratar ambas como apenas uma aresta não orientada. Estas arestas são adicionadas de maneira que seja gerada uma árvore dos caminhos mais curtos relativamente ao vértice que é gateway. Posteriormente, são acrescentadas aproximadamente $\frac{m \times n}{3}$ arestas, não orientadas, ao grafo gerado. Estas arestas são posicionadas de forma a ligarem vértices que, para comunicarem, tenham de percorrer uma distância superior à que existe fisicamente entre eles, ou seja, vamos adicionar "atalhos" à rede.

Implementámos um algoritmo em MATLAB que gerasse um grafo físico da forma anteriormente descrita. Os parâmetros de geração deste grafo são os seguintes: o número de vértices que pretendemos que o grafo físico tenha, isto é, o valor para n e m; e as capacidades que as arestas e os vértices, que neles têm localizados data centers, devem ter.

O grafo da Figura 4.4 foi obtido através do programa explicado anteriormente e é um exemplo de um grafo físico com 10×10 vértices. Este grafo ilustra uma rede física com 100 antenas.

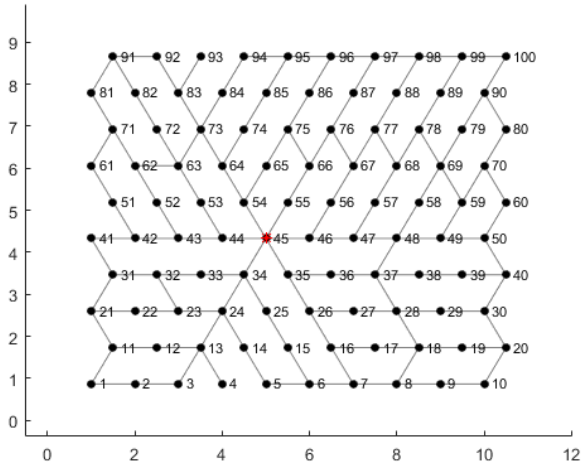


Fig. 4.4 Grafo que ilustra uma rede física

Tendo em conta o que já foi explicado anteriormente na Secção 4.2 sobre o grafo virtual e que estamos a considerar uma rede física com $m \times n$ antenas, escolhemos como modelo para o grafo virtual um grafo orientado que fosse composto por $m \times n$ cadeias. Cada uma destas cadeias representa o conjunto dos pedidos que são feitos a cada uma das antenas da rede física. Cada uma das cadeias, será constituída por três vértices. O primeiro vértice representa a antena que está ligada a um vértice intermédio que representa um data center. Por sua vez, este vértice intermédio, vai estar ligado a um terceiro vértice que representa o gateway que é comum a todas as cadeias. De forma a conseguirmos distinguir facilmente as funções de cada vértice, quando visualizamos o grafo gerado, decidimos colorir os mesmos com cores distintas. Através da Figura 4.5 conseguimos observar essa coloração onde os vértices que representam as antenas, os data centers e o gateway estão coloridos a azul, a preto e a vermelho, respetivamente.

Assim, implementámos um algoritmo em MATLAB que gerasse um grafo virtual da forma anteriormente descrita. Os parâmetros de geração deste grafo são os seguintes: o número de vértices que pretendemos que o grafo físico tenha, isto é, o valor para n e m; e os pesos que as arestas e os vértices que representam data centers devem ter.

No algoritmo implementado, definimos como âncoras o mapeamento do gateway no grafo virtual para o gateway no grafo físico. Definimos ainda que a antena i do grafo virtual deve ser mapeada para a antena i do grafo físico.

O grafo da Figura 4.5 foi obtido através do programa explicado anteriormente e é um exemplo de um grafo virtual, composto por 100 cadeias. Assim, este grafo é constituído por 100 antenas, 100 vértices intermédios e 1 gateway. Este grafo ilustra a rede virtual.

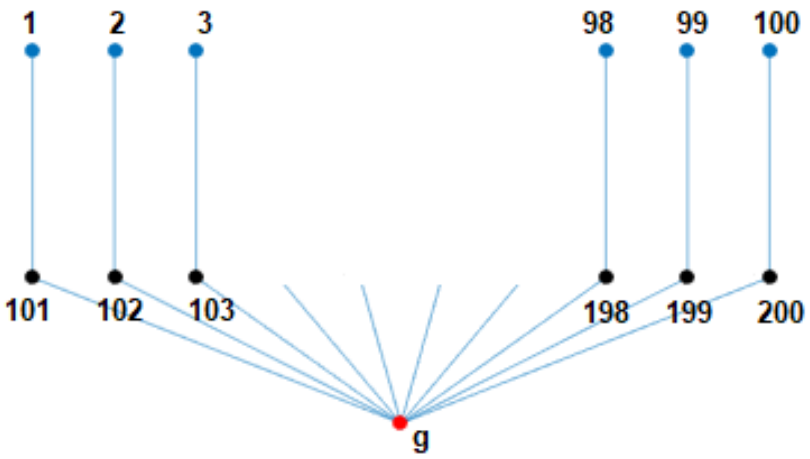


Fig. 4.5 Grafo que ilustra uma rede virtual

A implementação da formulação deste problema será análoga à elaborada na Secção 3.3. Isto significa que vamos recorrer novamente ao programa MATLAB para gerar os grafos e ao IBM ILOG CPLEX Optimization Studio para nos fornecer a solução do problema, através da implementação feita no mesmo.

A solução que obtivemos, neste teste em particular, teve em conta que pretendemos minimizar a distância a percorrer pelos dados a processar, sendo que damos mais importância à minimização entre antenas e vértices intermédios do que entre vértices intermédios e gateway. Assim, neste teste, consideramos f_2 (4.5) como função de minimização e atribuímos os seguintes valores às constantes da função: $d_1 = 3$ e $d_2 = 1$. Ainda neste teste, na restrição (4.2), consideramos a constante $c = 11$.

Correndo o programa implementado no IBM ILOG CPLEX Optimization Studio, obtemos como solução três matrizes. Tal como na Secção 3.3, duas destas matrizes representam as variáveis binárias $\triangle_i^k$ e $\triangle_{(i,j)}^{(k,l)}$. A terceira representa a variável binária $\triangle_i$, que nos indica, através dos seus números binários, que vértices do grafo físico têm neles localizados data centers, sendo que 1 indica que o vértice contém um data center e 0 caso contrário.

A Figura 4.6 foi conseguida através de um algoritmo implementado em MATLAB para tornar a solução visível. Esta figura mostra a solução que obtivemos para o mapeamento do grafo virtual da Figura 4.5 para o grafo físico da Figura 4.4. Na figura, as arestas utilizadas no mapeamento encontram-se coloridas a verde e as não utilizadas a preto. O vértice que representa o gateway encontra-se assinalado através de um quadrado delineado a vermelho. Ainda na mesma imagem podemos observar alguns vértices representados por um quadrado colorido e que se encontram dentro de um hexágono branco. Estes vértices são a localização dos data centers. A cor de cada hexágono permite-nos identificar qual o data center que está a ser utilizado por essa mesma antena. A figura mostra-nos a localização, na rede, dos 11 data centers.

Nos testes numéricos que realizámos, e que vamos apresentar de seguida, variámos o tamanho do grafo físico e, por consequência, o tamanho do grafo virtual. Isto porque, pelas razões explicadas anteriormente, se temos o primeiro com $m \times n$ vértices, então o segundo terá $2 \times m \times n + 1$ vértices.

Nas Tabelas 4.1 e 4.2 estão descritos, para cada teste que realizámos, o número de vértices que considerámos para o grafo físico e os tempos que o programa demorou até obter a solução para cada

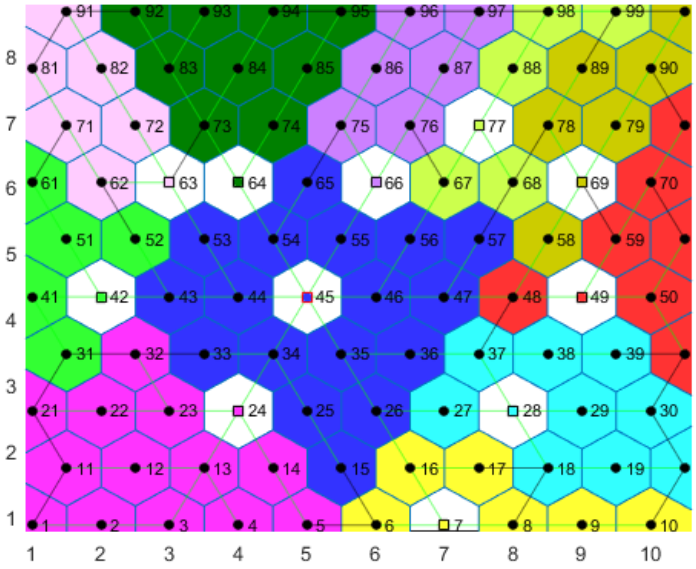


Fig. 4.6 Solução obtida

um deles. Por fim, na última coluna, estão descritos o número de data centers, localizados no grafo, segundo a solução que obtivemos.

Quando gerámos as instâncias para os testes, escolhemos um valor elevado para os parâmetros das capacidades, neste caso 100, e um valor baixo para os parâmetros dos pesos, neste caso 1. Considerámos estes valores para que os resultados dos testes numéricos não ficassem logo comprometidos pela falta de solução. Na prática, é como se não estivéssemos a considerar capacidades. O objetivo desta não consideração é tentarmos perceber se existirá outro fator que dificulte o mapeamento que não as capacidades serem excedidas.

Nos testes, que apresentamos na Tabela 4.1, não considerámos a restrição (4.2) pois queremos perceber como variam o número de data centers localizados relativamente ao número de vértices considerados para o grafo físico. Assim, nestes testes, considerámos a função f_4 como função de minimização com $d_1 = 3$, $d_2 = 1$, $w_1 = 0,95$ e $w_2 = 0,05$. As variáveis w_1 e w_2 somam o valor 1. Explicaremos os pesos com mais precisão mais adiante na Secção 4.4 quando abordarmos o problema multiobjetivo.

Da Tabela 4.1 podemos observar que o número de data centers localizados representa aproximadamente 10% relativamente ao número de vértices considerados para o grafo físico e aos parâmetros assumidos. Em todos os 9 casos testados, foi possível obter uma solução, sendo que o último caso testado demorou bastante tempo até apresentar a mesma, em comparação com os outros testes. Isto já seria espectável face ao tamanho considerado para o grafo físico nesse mesmo teste.

Nos testes, que apresentamos na Tabela 4.2, passámos a considerar a restrição (4.2), ou seja, passámos a ser nós a determinar o número de data centers a localizar no grafo. A coluna mais à direita da tabela indica o valor que impusemos para os mesmos. Assim, nestes testes, considerámos a função f_2 como função de minimização com $d_1 = 3$ e $d_2 = 1$.

Relativamente aos 9 casos testados, descritos na Tabela 4.2, foi possível obter uma solução em todos, novamente, sendo que o último caso foi também o que demorou muito mais tempo relativamente aos outros testes. Novamente era de prever tendo em conta o tamanho considerado para o grafo físico.

Nº do teste	Nº vértices do grafo de chegada	Tempo (Segundos)	Nº Data centers
1	$3 \times 4 = 12$	0,072	1
2	$3 \times 7 = 21$	1,078	1
3	$7 \times 3 = 21$	1,063	2
4	$5 \times 5 = 25$	1,065	1
5	$8 \times 9 = 72$	12,058	7
6	$10 \times 12 = 120$	35,025	13
7	$12 \times 10 = 120$	35,056	13
8	$15 \times 15 = 225$	248,002	24
9	$20 \times 20 = 400$	7800,038	48

Tabela 4.1 Tabela de testes sem a restrição (4.2)

Nº do teste	Nº vértices do grafo de chegada	Tempo (Segundos)	Nº Data centers
10	$3 \times 4 = 12$	1,015	2
11	$3 \times 7 = 21$	1,032	4
12	$5 \times 5 = 25$	1,033	1
13	$5 \times 5 = 25$	1,047	3
14	$8 \times 9 = 72$	15,033	7
15	$12 \times 10 = 120$	52,073	13
16	$12 \times 10 = 120$	37,058	20
17	$15 \times 15 = 225$	499,083	25
18	$20 \times 20 = 400$	10210,093	50

Tabela 4.2 Tabela de testes com a restrição (4.2)

4.4 Fronteira Pareto-ótima

No problema da localização de processadores na periferia da rede existem vários fatores que se devem ter em conta. Um deles, já mencionado, é a localização de uma quantidade ótima de data centers na rede. Isto deve-se ao facto de os data centers não poderem ser colocados nem em demasia, visto que torna o processo caro, nem em insuficiência, sob pena de colocar em causa o serviço prestado pela rede.

Outra situação que se deve ter em conta é que a passagem de dados na rede deve percorrer a menor distância possível, respeitando as ligações existentes entre as antenas. Estes são apenas dois exemplos, entre muitos, de fatores que se podem ter em conta no nosso problema.

Assim, poderá ser útil, em algumas circunstâncias, abordar o problema da localização de processadores na periferia da rede como sendo um problema de otimização multiobjetivo. Aqui temos como objetivo uma minimização de vários critérios em simultâneo. Em particular, se se considerarem apenas dois critérios em simultâneo, estamos perante um problema biobjetivo. Os critérios tidos em

conta podem ser conflitantes entre si, isto é, a melhoria de algum deles pode causar a piora do outro. Conciliar este facto torna o problema mais interessante na nossa perspetiva.

Desta forma, podemos formular um problema de otimização multiobjetivo da seguinte maneira:

$$\begin{aligned} &\min\ (\max)\ \ z = (f_1(x), f_2(x), \ldots, f_n(x)) \\ &s.a.\ \ x \in X^* \end{aligned} \tag{4.8}$$

onde z é o vetor das funções objetivo e X^* é o conjunto de soluções do problema.

Num problema deste tipo, quando consideramos objetivos conflitantes, não conseguimos obter uma só solução ótima relativamente a todos os objetivos tidos em conta em simultâneo. Desta forma, o nosso objetivo será encontrar o *conjunto Pareto-ótimo*, que é o conjunto de todas as *soluções não-dominadas* de entre as possíveis. As soluções que formam este conjunto designam-se por *soluções eficientes* ou *soluções Pareto-ótimas*. Este conjunto de soluções é possível de determinar através da *dominância de Pareto*.

Dizemos então $x \in X^*$ é uma solução possível, que se designa *Pareto-ótima*, caso não exista uma outra solução $x' \in X^*$ tal que x' não seja pior que x em nenhum objetivo e x' seja estritamente melhor que x em pelo menos um objetivo. Isto traduz-se pelas condições seguintes.

$$f_k(x') \leq f_k(x) \quad \forall k = 1,2,\ldots,n \qquad \wedge \qquad \exists j \in \{1,2,\ldots,n\} \quad f_j(x') < f_j(x). \tag{4.9}$$

Caso exista uma solução x' que verifique a condição (4.9), então a solução x é dominada pela solução x' o que significa que x não fará parte do conjunto Pareto-ótimo. Caso exista um i e um j tal que

$$f_j(x') < f_j(x) \qquad \wedge \qquad f_i(x') > f_i(x) \tag{4.10}$$

então as soluções x e x' são chamadas *soluções não-dominadas entre si* ou *indiferentes*.

A Figura 4.7 é um exemplo para o conceito de dominância de Pareto. Neste caso é utilizada a solução x como referência para um problema de minimização das funções f_1 e f_2.

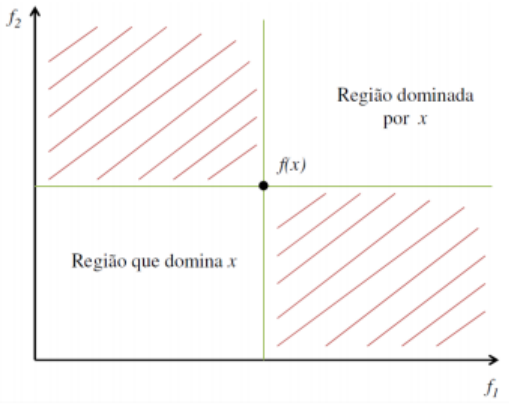


Fig. 4.7 Conceito de dominância de Pareto

Denominamos o conjunto imagem do conjunto Pareto-ótimo por *fronteira Pareto-ótima*.

Escolhemos o *Método de Ponderação dos Objetivos* para resolver o problema. Este método permite-nos substituir o problema de otimização multiobjetivo por um de otimização escalar através de uma função mono-objetivo. Esta função será a soma ponderada dos vários objetivos que vamos considerar. Temos então a seguinte função mono-objetivo:

$$f(x) = \sum_{i=1}^{n} w_i f_i(x) \tag{4.11}$$

onde $w_i \geq 0$ são os pesos ou coeficientes de ponderação. Estes pesos exprimem a importância relativa de cada função-objetivo f_i. Tipicamente assumimos que:

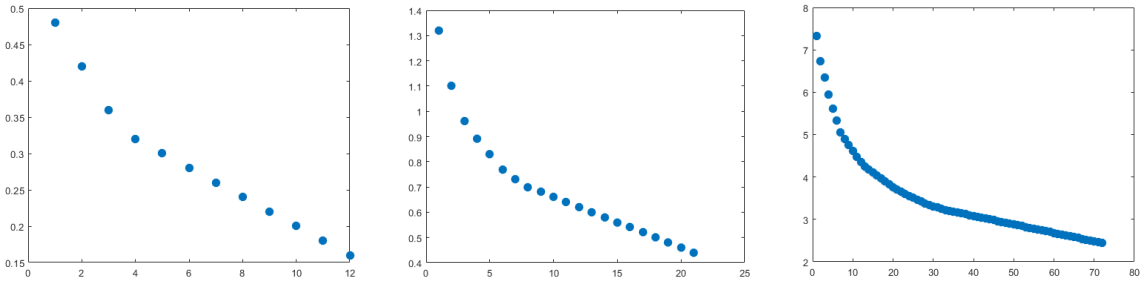
$$\sum_{i=1}^{n} w_i = 1 \tag{4.12}$$

Note-se que encontrar pesos adequados para a função em causa pode ser complicado. Uma forma de contornar esta situação será irmos atribuindo diferentes valores a w_i.

Mais sobre programação linear multiobjetivo pode ser encontrado em [6].

De seguida, vamos apresentar alguns exemplos de fronteira de Pareto-ótima que obtivemos para alguns dos testes numéricos que apresentámos na Secção 4.3. As considerações para os grafos e para a função objetivo a minimizar foram as mesmas que utilizámos para os testes numéricos da Secção 4.3.

A Figura 4.8a ilustra a fronteira de Pareto-ótima para o teste 1 e 10 da secção mencionada anteriormente. Da mesma forma, a Figura 4.8b corresponde aos testes 2, 3 e 11 e a Figura 4.8c aos testes 5 e 14.



(a) Grafo físico com 12 vértices (b) Grafo físico com 21 vértices (c) Grafo físico com 72 vértices

Fig. 4.8 Fronteira Pareto-ótima

Capítulo 5

Problema da imersão de redes considerando padrões de mobilidade

Neste capítulo vamos apresentar o problema da imersão de redes considerando padrões de mobilidade. Vamos ainda apresentar alguns testes numéricos que realizamos de forma a percebermos como é que os padrões de mobilidade dos utilizadores influenciam o resultado. Por fim, aplicamos este modelo a um caso realista.

5.1 Problema da imersão de redes considerando padrões de mobilidade

O problema da imersão de redes considerando padrões de mobilidade (Mobility Aware Virtual Network Embedding problem) é uma extensão do problema da imersão de redes. O estudo deste problema é motivado por investigações que revelam que o efeito da mobilidade dos utilizadores tem um impacto significativo no procedimento de imersão de pedidos numa rede física.

Na realidade os utilizadores numa rede móvel deslocam-se, podendo passar da zona afeta a uma antena para uma zona afeta a outra, com possível alteração de data center. Esta mudança implica a transferência de dados entre os dois data centers em questão.

Na Figura 5.1 podemos observar um esquema simplificado da arquitetura de rede com mobilidade.

Nesta extensão que estamos a abordar, apenas a estrutura ou rede virtual, isto é, a abstração dos pedidos feitos à estrutura ou rede física, vão refletir o efeito da mobilidade dos utilizadores.

Tal como já referimos, a mobilidade dos utilizadores pode implicar uma transferência de dados do data center a que o mesmo se encontra afeto para um outro que sirva a nova antena para o qual se deslocou. Para capturar esta transferência de dados no modelo, as cadeias da estrutura virtual vão ter de estar ligadas entre si, ao nível dos data centers, se existir transferência de utilizadores entre as antenas associadas. Assim, estas novas ligações que necessitamos, são representadas, no grafo virtual, por arestas que ligam data centers, de cadeias distintas, cujas respetivas antenas sejam vizinhas. As antenas têm de ser vizinhas porque só se pode transferir dados entre antenas que o sejam.

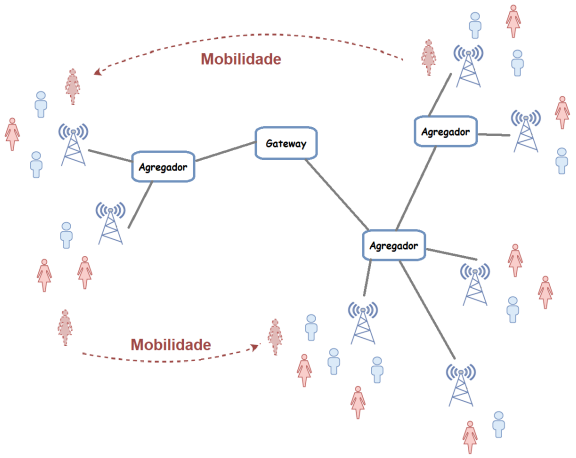


Fig. 5.1 Esquema da arquitetura da rede com mobilidade

O grafo da Figura 5.2 é um exemplo de um grafo virtual considerando padrões de mobilidade. Nela podemos observar estas novas ligações, entre data centers, ou seja, entre os vértices intermédios coloridos a preto.

Podemos observar pela Figura 5.2 que, na visualização do grafo gerado, continuamos a distinguir as antenas, os vértices intermédios e o gateway através da cor azul, preta e vermelha, respetivamente.

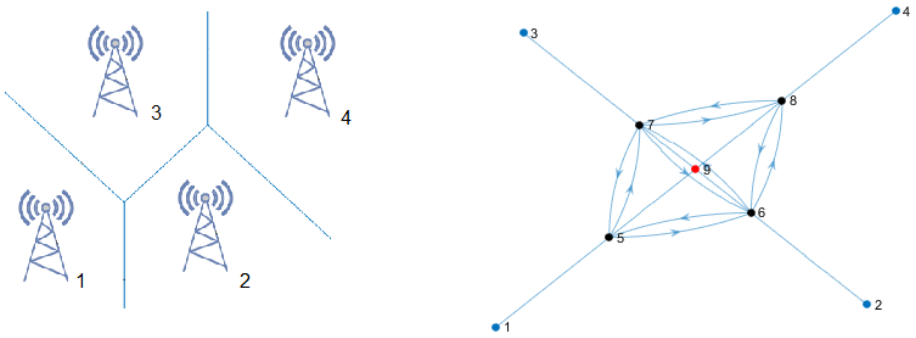


Fig. 5.2 Distribuição de antenas numa área geográfica e grafo que ilustra a rede virtual correspondente considerando padrões de mobilidade

Tal como as ligações já existentes, também cada uma destas novas ligações têm um peso associado. O peso associado a estas novas ligações reflete a possibilidade de haver transferência de dados entre antenas vizinhas. Sendo em geral proporcional a quantidade de procura transferida. A constante de proporcionalidade depende da aplicação concreta.

Por exemplo, no caso da Figura 5.2 a antena 1 tem como vizinhas, e por isso como opções de mobilidade, as antenas 2 e 3. Não existe possibilidade de haver transferência de dados entre a antena 1 e a 4, pelo menos de forma direta.

No estudo deste problema, vamos continuar a pretender localizar os processadores da periferia da rede, os data centers, agora considerando padrões de mobilidade e vamos continuar a tratar o problema como sendo um problema multiobjetivo. Desta forma, a formulação deste problema será igual à formulação do problema da imersão de redes explicada na Secção 4.2.

Mais sobre o problema da imersão de redes virtuais considerando padrões de mobilidade pode ler-se no artigo [5].

5.2 Testes numéricos considerando padrões de mobilidade

Vamos começar por gerar instâncias de testes artificiais baseadas no mesmo caso simplificado usado na Secção 4.3. Apenas vamos explicar a nova geração do grafo virtual pois foi este o único a ser alterado de acordo com o explicado acima. Vamos também explicar como foi implementado o problema para que, de seguida, possamos apresentar alguns testes numéricos que realizámos.

Na geração do grafo virtual considerando padrões de mobilidade, vamos manter, na sua essência, a geração explicada na Secção 4.3 para o grafo virtual sem estes padrões. Assim, continuamos a pretender que o grafo virtual seja composto por $m \times n$ cadeias. Cada uma destas cadeias continua a ser composta por três vértices, uma antena ligada a um vértice intermédio que, por sua vez, está ligado a um gateway. Continuamos a considerar apenas um gateway comum a todas as cadeias, visto que o grafo físico se mantém com apenas um gateway também.

Os pesos que vamos associar às arestas entre vértices intermédios são calculados em função do número de vizinhos de cada antena e definimos que o peso associado à mudança de antena é de 0,05.

Como estamos a considerar movimentações homogêneas, a procura que sai da antena i para a j é compensada pela que vai da j para a i pelo que consideramos ainda o peso das arestas entre antenas e vértices intermédios e entre vértices intermédios e o gateway sempre iguais a 1.

Implementámos um algoritmo em MATLAB que gerasse um grafo virtual tal como descrevemos anteriormente.

Continuamos a definir como âncoras o mapeamento do gateway no grafo virtual para o gateway no grafo físico. Também definimos que a antena i do grafo virtual tem de ser mapeada para a antena i do grafo físico. Não impomos âncoras para os vértices intermédios visto que será a solução que determinará a localização dos mesmos.

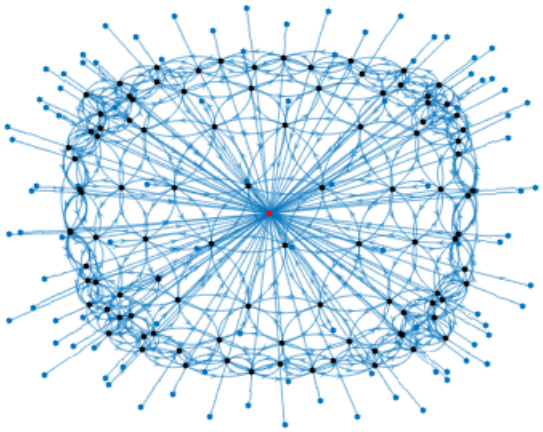


Fig. 5.3 Grafo que ilustra uma rede virtual de dimensão 100, considerando padrões de mobilidade

O grafo da Figura 5.3 foi obtido através do programa explicado anteriormente e é um exemplo de um grafo virtual considerando padrões de mobilidade composto por 100 cadeias, isto é, 100 antenas, 100 data centers e 1 gateway.

A implementação do problema da imersão de redes considerando padrões de mobilidade, será análoga à implementação descrita para o problema da imersão de redes. Assim, no MATLAB começamos por gerar os dois grafos, o físico e o virtual. De seguida, recorremos ao programa IBM ILOG CPLEX Optimization Studio, onde implementámos o problema formulado, com o intuito de obtermos uma solução para o mesmo.

A solução que nos é fornecida teve em conta que pretendemos minimizar a distância a percorrer pelos dados a processar. Agora, como estamos a considerar arestas entre vértices intermédios no grafo virtual, vamos ter de acrescentar mais variáveis à função de minimização. Desta forma, obtemos uma nova função de minimização (5.1) para o problema.

$$f_5 = \sum_{(k,l)\in E_G,(n,m)\in E_H} \triangle_{(n,m)}^{(k,l)} \left[d_1 \sum_{k\in V_A, l\in V_D} p_{(k,l)} d_{(n,m)} + d_3 \sum_{k,l\in V_D} p_{(k,l)} d_{(n,m)} + d_2 \sum_{l\in V_G, k\in V_D} p_{(k,l)} d_{(n,m)} \right] \tag{5.1}$$

Como solução do problema obtemos três matrizes de adjacência, isto é, obtemos as soluções para as variáveis binárias $\triangle_i^k$ e $\triangle_{(i,j)}^{(k,l)}$ e $\triangle_i$. Com a solução determinada, recorremos ao MATLAB, onde criamos outro programa para tornar a solução visível.

Na figura que ilustra a solução, o vértice que representa o gateway continua a destacar-se através de um quadrado a vermelho. Podemos ainda visualizar, na figura, alguns vértices representados por um quadrado, colorido no seu interior. Estes vértices identificam os data centers. A cor de cada hexágono permite-nos identificar qual o data center que está a ser utilizado por uma determinada antena. As arestas que foram utilizados no mapeamento encontram-se coloridas a verde e as que não foram a preto.

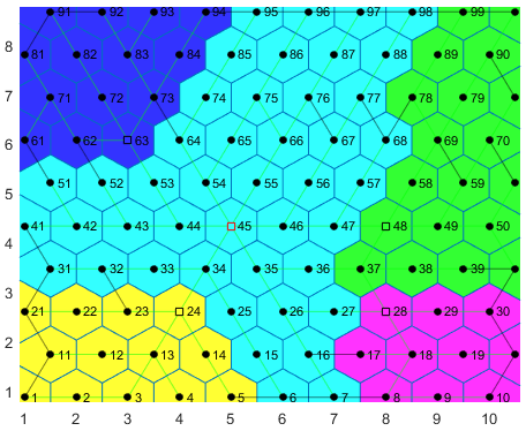


Fig. 5.4 Solução obtida com a restrição (4.2)

A Figura 5.4 foi conseguida através dos programas explicados anteriormente e mostra a solução obtidas para o mapeamento do grafo virtual da Figura 5.3 para o grafo físico da Figura 4.4.

Para obtermos a Figura 5.4 impusemos a restrição (4.2) com $c = 5$ e considerámos os seguintes valores para as variáveis da função (5.1): $d_1 = 2$, $d_2 = 1$ e $d_3 = 2$.

Para que consigamos realmente observar os padrões de mobilidade vamos simular que no grafo físico existe uma zona onde é mais frequente o utilizador mover-se. Vamos assumir que uma estrada é uma zona propícia à ocorrência de mobilidade. Desta forma, vamos simular no grafo físico uma estrada através de uma reta. Esta reta vai passar por diferentes locais, onde cada um deles estará abrangido por uma determinada antena. Ao considerarmos esta reta, como sendo uma zona propícia à ocorrência de mobilidade, os pesos das arestas que conectam os vértices intermédios vão sofrer alterações face aos valores que foram estabelecido anteriormente para os mesmos.

Considerámos dois casos distintos para essa alteração dos pesos. Ambos os casos serão abordados e analisados de seguida.

Caso 1 *Se a estrada, isto é, a reta, passar por uma zona que é abrangida pela antena i, adiciona-se peso 2 ao peso da aresta que liga a antena i ao respetivo vértice intermédio e ao peso da aresta que liga esse mesmo vértice intermédio ao gateway. Todos os pesos das outras arestas mantêm-se inalterados.*

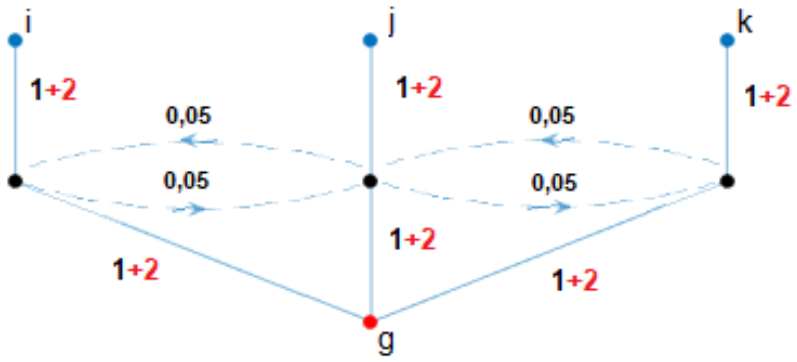


Fig. 5.5 Ilustração do Caso 1 quando i, j e k são antenas atravessadas sucessivamente por uma estrada

Caso 2 *Se a estrada, isto é, a reta, passar por uma zona que é abrangida pelas antenas i e j, com i e j antenas vizinhas, adiciona-se peso 2 ao peso da aresta que liga a antena i e j ao respetivo vértice intermédio e ao peso da aresta que liga esse mesmo vértice intermédio ao gateway. Adiciona-se também peso 1 ao peso da aresta que liga o vértice intermédio da cadeia da antena i ao vértice intermédio da cadeia da antena j. Da mesma forma, adiciona-se peso 1 ao peso da aresta que liga o vértice intermédio da cadeia da antena j ao vértice intermédio da cadeia da antena i. Todos os pesos das outras arestas mantêm-se inalterados.*

O Caso 1, Figura 5.5, corresponde essencialmente a aumentar a procura na estrada, mas considerando que a nova procura não se move. O Caso 2, Figura 5.6, corresponde ao caso extremo em que adicionamos a mesma procura mas assumimos que toda ela se vai mover para uma das antenas adjacentes.

De forma a conseguirmos distinguir melhor os dois casos, denominamos o Caso 1 por *caso parado* e o Caso 2 por *caso movimento*.

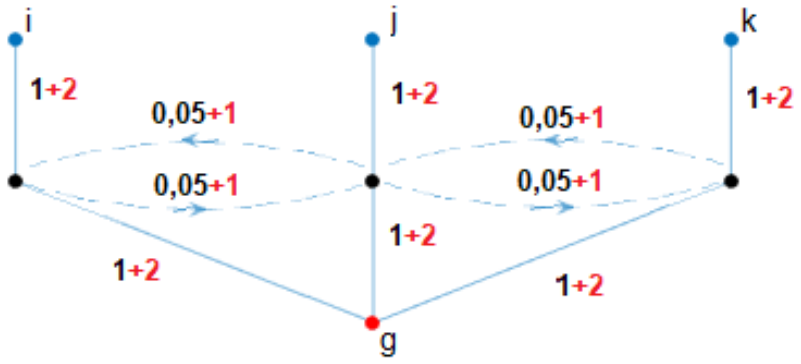


Fig. 5.6 Ilustração do Caso 2 quando i, j e k são antenas atravessadas sucessivamente por uma estrada

Nos testes que vamos apresentar de seguida, com o intuito de aferirmos o problema formulado, vamos considerar como rede física o grafo ilustrado na Figura 4.4 e como rede virtual o grafo ilustrado na Figura 5.3. Com estes testes pretendemos também tentar visualizar o efeito dos padrões de mobilidade que considerámos.

Nestes teste testamos os dois casos explicados anteriormente, o caso parado e o caso movimento. Em todos eles impusemos a restrição (4.2), onde o parâmetro c assumiu o valor 5, 10 ou 15. Isto significa que, em cada teste que realizámos, impusemos que o grafo tivesse 5, 10 ou 15 data centers. Com esta imposição, resta ao programa, que determina a solução, localizá-los.

Para realizarmos estes testes, assumimos que o grafo que estamos a utilizar se encontra numa zona de procura reduzida, daí que as capacidades não sejam importantes. Por esta razão é que considerámos elevadas capacidades e baixos pesos. Na prática é como se não estivéssemos a utilizar capacidades nos testes. Assim, continuamos a considerar o valor 100, valor elevado, para o parâmetro das capacidades dos vértices e das arestas do grafo físico. Para o grafo virtual, também continuamos a considerar o valor 1, valor baixo, para os pesos dos vértices. Em particular, os pesos das arestas do grafo virtual são determinadas consoante o caso que estejamos a considerar, o caso parado ou o caso movimento.

Definimos, na função de minimização (5.1), que $d_1 = 2$, $d_2 = 1$ e vamos variar o parâmetro d_3 de forma a percebermos a importância das arestas entre data centers para retratar o efeito dos padrões de mobilidade. Assim, o parâmetro d_3 vai assumir, em testes diferentes, os valores 2 e 1.

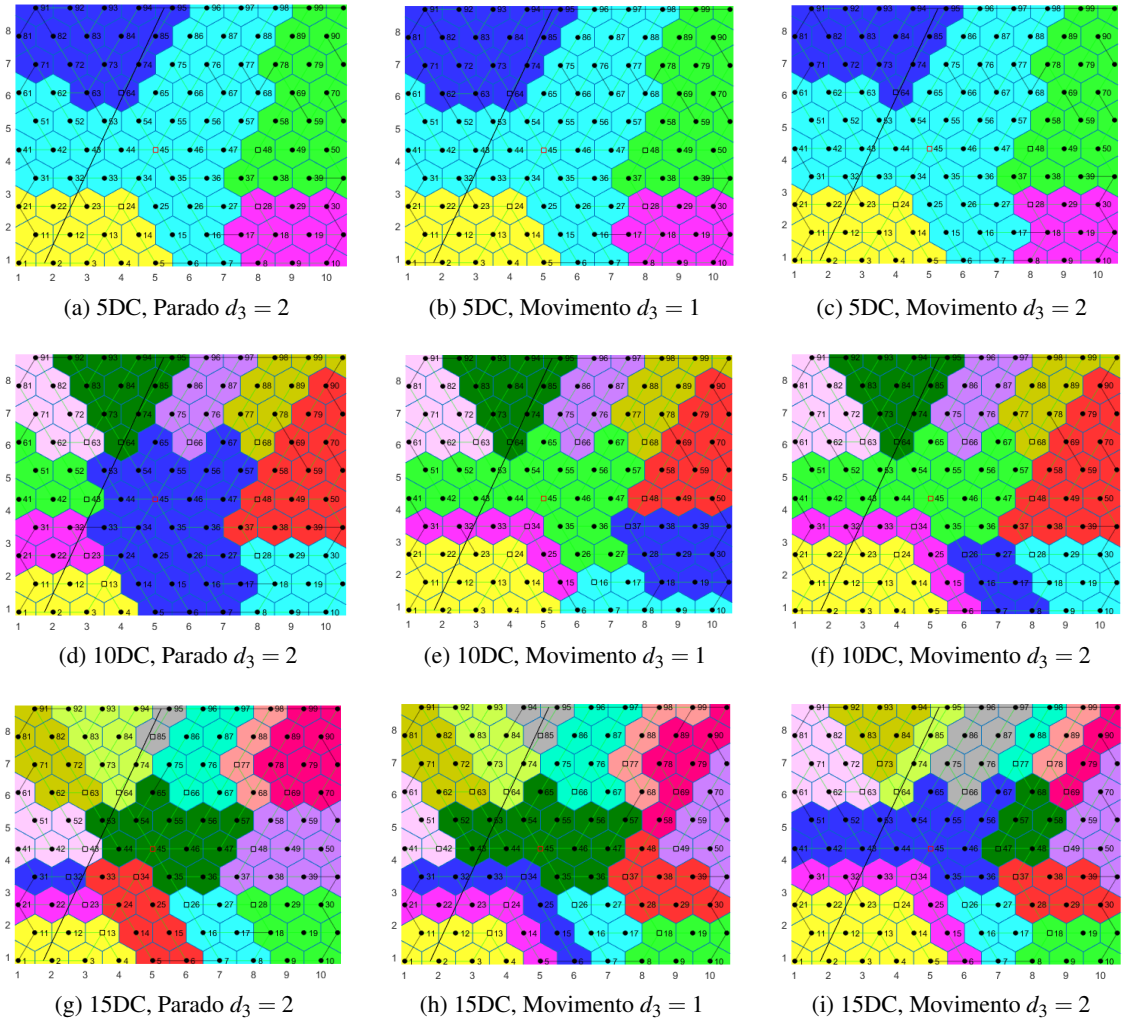
Com o intuito de conseguirmos visualizar o efeito dos padrões de mobilidade em função da localização da zona propícia à ocorrência de mobilidade, decidimos simular três estradas distintas.

Estrada	Nº Data centers	Parado $d_3 = 2$	Movimento $d_3 = 1$	Movimento $d_3 = 2$
1	c=5	319,010 Seg.	372,049 Seg.	286,016 Seg.
1	c=10	197,056 Seg.	192,086 Seg.	149,045 Seg.
1	c=15	152,097 Seg.	114,074 Seg.	116,098 Seg.
2	c=10	177,052 Seg.	187,055 Seg.	220,016 Seg.
3	c=10	154,085 Seg.	207,053 Seg.	162,002 Seg.

Tabela 5.1 Tabela de testes considerando os padrões de mobilidade

Na Tabela 5.1 estão registados os tempos que cada teste demorou até apresentar a solução do mesmo. Tendo em conta a dimensão dos grafos utilizados, esperávamos que os tempos para obtermos as diferentes soluções, fossem superiores aos que podemos observar na tabela mencionada. Como observamos, quanto menor for o valor escolhido para o parâmetro c maior será o tempo até obtermos a solução.

Vamos então analisar as Figuras 5.7, 5.8 e 5.9 que são as soluções obtidas para os diferentes testes.



(a) 5DC, Parado $d_3 = 2$ (b) 5DC, Movimento $d_3 = 1$ (c) 5DC, Movimento $d_3 = 2$

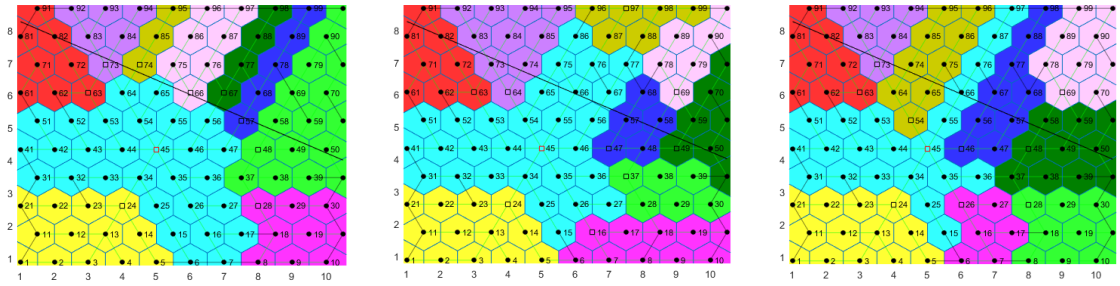
(d) 10DC, Parado $d_3 = 2$ (e) 10DC, Movimento $d_3 = 1$ (f) 10DC, Movimento $d_3 = 2$

(g) 15DC, Parado $d_3 = 2$ (h) 15DC, Movimento $d_3 = 1$ (i) 15DC, Movimento $d_3 = 2$

Fig. 5.7 Testes com mobilidade estrada 1

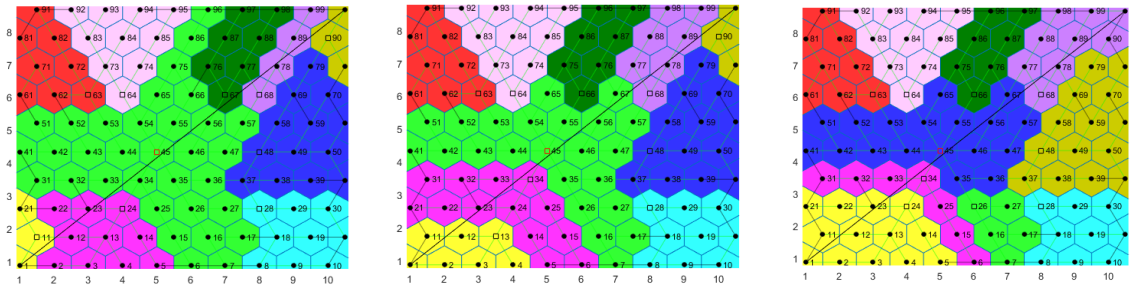
A Figura 5.7, que têm 9 sub figuras, são as soluções obtidas considerando a estrada número 1 para o efeito da mobilidade. Analogamente, as Figuras 5.8 e 5.9, que têm cada uma 3 sub figuras, são as soluções obtidas considerando a estrada número 2 e número 3, respetivamente.

Comecemos por comparar as imagens da esquerda, que correspondem ao caso parado, com as do meio e as da direita, que correspondem ao caso movimento. No caso parado, há mais data centers a abranger a estrada do que no caso movimento, independentemente do valor considerado para o parâmetro d_3. Este facto observa-se facilmente se contarmos o número de cores distintas que a estrada abrange em cada um dos casos.



(a) 10DC, Parado $d_3 = 2$ (b) 10DC, Movimento $d_3 = 1$ (c) 10DC, Movimento $d_3 = 2$

Fig. 5.8 Testes com mobilidade estrada 2



(a) 10DC, Parado $d_3 = 2$ (b) 10DC, Movimento $d_3 = 1$ (c) 10DC, Movimento $d_3 = 2$

Fig. 5.9 Testes com mobilidade estrada 3

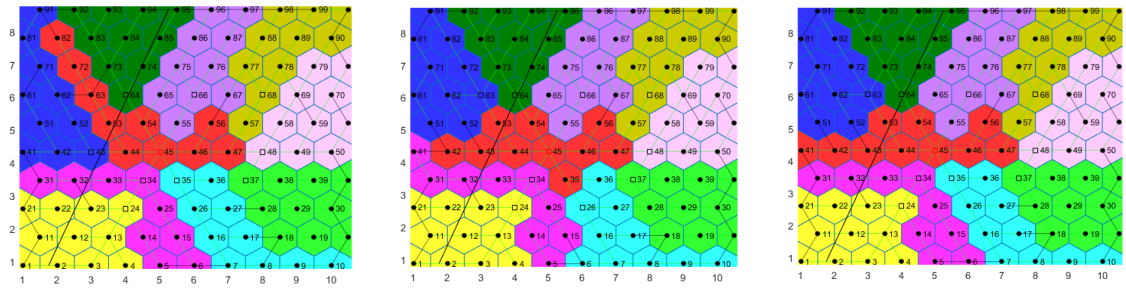
Comparemos agora as duas colunas mais à direita. Estes dois tipos de testes distinguem-se pelo facto de num deles darmos mais importância às arestas que ligam os data centers do que no outro. Daí que, no caso da direita, se atribua o valor 2 ao parâmetro d_3 enquanto que no do meio se atribui o valor 1.

Comparar os testes em relação às diferentes estradas consideradas, não é muito significativo. A única coisa que acontece com a mudança do tipo de estrada é que o posicionamento dos data centers é alterado em função das zonas por onde a estrada passa.

Com o intuito de percebermos a importância das capacidades no problema, decidimos fazer alguns testes impondo as mesmas. Assim, fomos testar os mesmos casos considerados anteriormente para a estrada 1 e onde impusemos a restrição que obriga a localização de 10 data centers. As 3 sub figuras da Figura 5.10 ilustram as soluções obtidas para estes testes. Definimos que cada aresta da rede física tinha 50 de capacidade e que cada vértice data center da mesma rede tinha capacidade 10.

Como definimos que cada antena que nela localize um data center, no grafo virtual, tinha peso 1 então ao impormos a capacidade 10 sabemos que, na solução, não pode haver mais que 10 antenas distintas associadas a um mesmo data center. Assim, ao considerarmos um grafo com 100 antenas, a solução teria de ter exatamente 10 antenas distintas associadas a cada um dos data centers. As capacidades vão limitar ainda mais a solução do problema como seria de esperar.

A disposição de cada antena em cada um dos testes, relativamente ao data center correspondente, é feita de forma a respeitar a maior ou menor importância que damos à proximidade entre data centers através do parâmetro d_3.

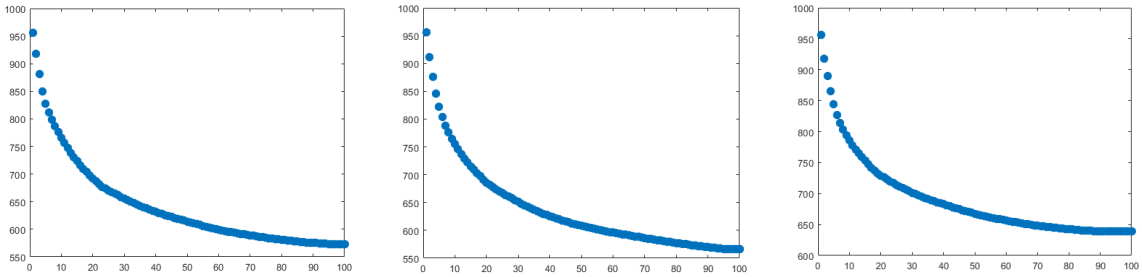


(a) 10DC, Parado $d_3 = 2$ (b) 10DC, Movimento $d_3 = 1$ (c) 10DC, Movimento $d_3 = 2$

Fig. 5.10 Testes com mobilidade e capacidades estrada 1

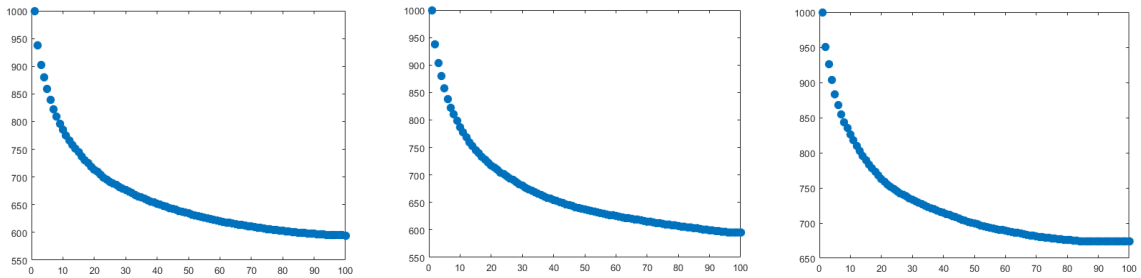
Podemos ainda, uma vez mais, usar otimização multiobjetivo em que deixamos de considerar restrição (4.2) e passamos a ter como objetivo de minimização f_5.

Nas Figuras 5.11, 5.12 e 5.13 podemos visualizar as diferentes fronteiras de Pareto obtidas para cada um dos 9 casos que considerámos.



(a) Parado $d_3 = 2$ (b) Movimento $d_3 = 1$ (c) Movimento $d_3 = 2$

Fig. 5.11 Fronteira de Pareto estrada 1

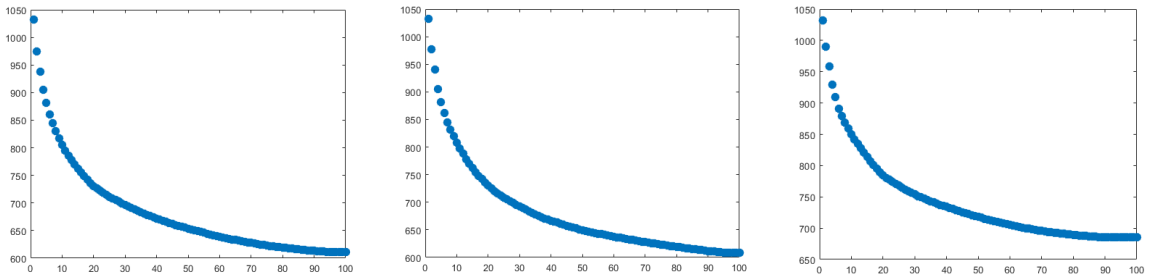


(a) Parado $d_3 = 2$ (b) Movimento $d_3 = 1$ (c) Movimento $d_3 = 2$

Fig. 5.12 Fronteira de Pareto estrada 2

Como estamos a considerar que o grafo físico é constituído por 100 antenas então podem ser localizados no grafo entre 1 e 100 data centers. Por esta razão, em cada uma das 9 sub figuras estão 100 pontos desenhados. Assim, o eixo dos xx representa o número de data centers.

Observando as figuras, concluímos que de facto quanto maior for o número de data centers associamos à rede menor serão os valores da função objetivo. Contudo, aumentar o valor da função



(a) Parado $d_3 = 2$ (b) Movimento $d_3 = 1$ (c) Movimento $d_3 = 2$

Fig. 5.13 Fronteira de Pareto estrada 3

objetivo vai implicar a redução do número de data centers e isto vai prejudicar a rede, tornando-a mais sobrecarregada e dando pior qualidade de tráfego ao utilizador. Assim, devemos tentar encontrar os valores que proporcionam o equilíbrio das circunstâncias.

5.3 Aplicação do problema a um caso realista

Nesta aplicação do problema a um caso realista, vamos considerar, como país, o Brasil. Tal como podemos observar pelos pontos azuis na Figura 5.14, este país é detentor de milhares de antenas, mais concretamente 93782.

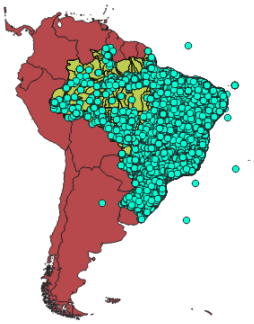
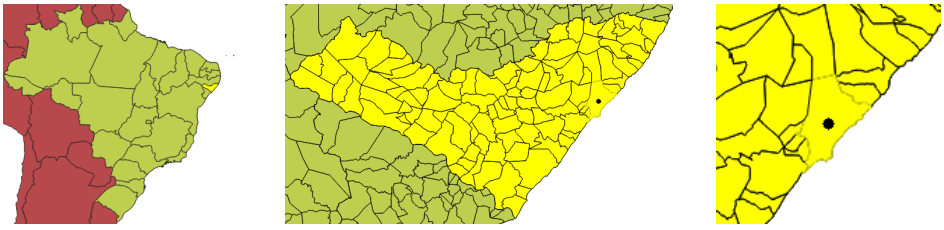


Fig. 5.14 Localização das antenas do Brasil

O Brasil encontra-se dividido por estados e cada estado por municípios. Como o número de antenas de todo o território em causa é muito elevado, decidimos restringir o território a considerar. Para isso, vamos restringir os dados relativos às antenas e considerar apenas o município de Maceió que se localiza no estado de Alagoas. Na Figura 5.15 podemos observar a localização da zona que vamos considerar para a aplicação do problema. Em particular, na Figura 5.15a, o estado de Alagoas encontra-se assinalado a amarelo e nas Figuras 5.15b e 5.15c, o município de Maceió está sinalizado com um ponto a preto.

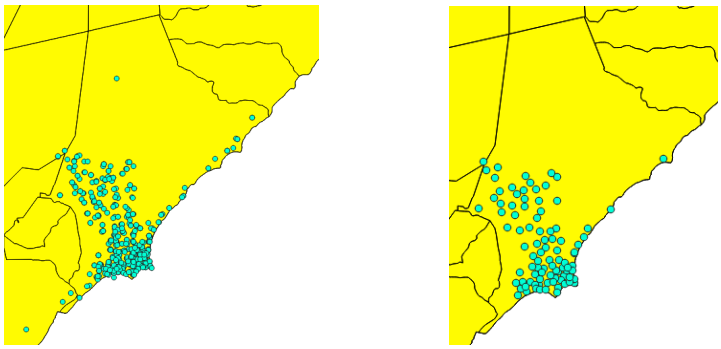
Na Figura 5.16a podemos observar, através dos pontos azuis, a localização de todas as antenas que o município Maceió detém. No total, Maceió tem 504 antenas. Estas 504 antenas não são geridas todas pela mesma operadora. Em Maceió operam 5 operadoras, Claro S.A., Nextel Telecomunicações



(a) Brasil (b) Estado Alagoas (c) Município Maceió

Fig. 5.15 Localização Brasil-Alagoas-Maceió

LTDA, Oi Móvel S.A., Telefônica Brasil S.A. e Tim S/A. Cada uma delas opera 92, 14, 146, 105 e 147 antenas, respetivamente.



(a) em Maceió (b) da operadora Claro S.A. em Maceió

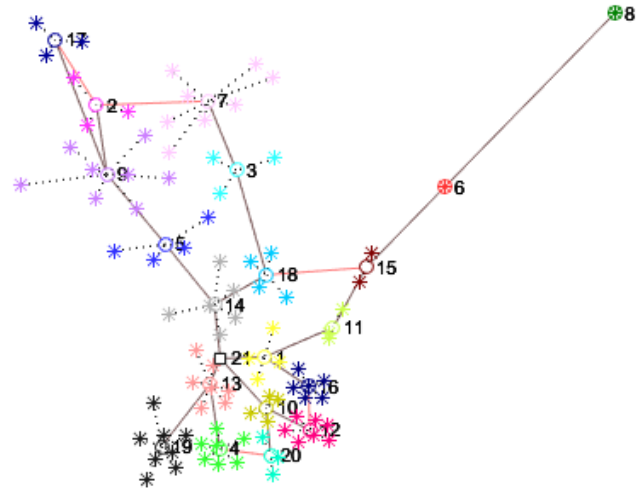
Fig. 5.16 Localização das antenas

Neste caso, vamos tratar os dados relativos às 92 antenas que a Claro S.A. detém. As localizações destas podem ser observadas na Figura 5.16b através dos pontos azuis. Para conseguirmos gerar a rede física do problema, necessitamos de guardar a informação relativa às coordenadas de cada uma das 92 antenas. Para isso, recorremos ao programa R que vai aceder a um ficheiro .csv com a informação relativa a todas as antenas do Brasil. Daqui, o programa vai selecionar as que pretendemos através do código lá implementado[1] para o efeito. Este programa vai devolver um outro ficheiro .csv com a informação das coordenadas das 92 antenas.

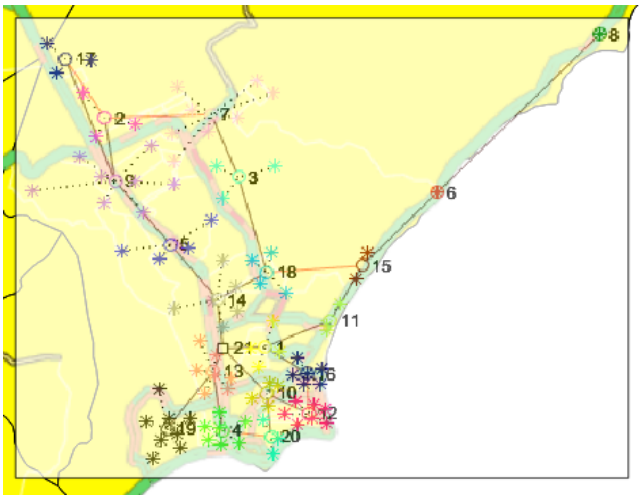
De seguida, recorremos ao MATLAB, com o intuito de recuperar o ficheiro gerado com as coordenadas das 92 antenas. Estas coordenadas vão permitir calcular as distâncias entre antenas e localizar as mesmas, ou seja, localizar os vértices, no nosso grafo que representa a rede física. Também neste programa, é gerado um número predefinido de pontos usando o algoritmo de agrupamento das médias-K *(K-means)* com o intuito de estes representarem agregadores. Por fim, é gerado um outro ponto, com uma localização predefinida para ilustrar o gateway. Assim, o grafo que ilustra a rede física tem como vértices o conjunto de pontos das antenas, dos agregadores e do gateway. As arestas do grafo físico são geradas de forma um pouco diferente da explicada na Secção 5.2. Todas as antenas

[1] O ficheiro .csv e o código implementado no programa R foram cedidos pelo aluno de doutoramento do DEI, David Lima, a quem agradecemos.

estão ligadas a um agregador, sendo que um agregador pode receber mais do que uma antena. Desta forma, cada uma das antenas vai estar ligada, através de uma aresta, ao vértice mais próximo que represente um agregador. Após a geração destas arestas, são geradas mais arestas de forma a criar uma árvore dos caminhos mais curtos entre os agregadores e o gateway. Por fim, são acrescentadas aproximadamente $\frac{k}{4}$ arestas, com k a representar o número predefinido de agregadores. Estas arestas, que estamos a acrescentar, são posicionadas de forma a ligarem agregadores que, para comunicarem entre si ou com o gateway, tenham de percorrer uma distância superior à que existe fisicamente entre eles, ou seja, vamos adicionar "atalhos" à rede. É de salientar que todas as arestas que compõem o grafo são não orientadas.



(a) rede física



(b) Sopreposição da rede física

Fig. 5.17 Rede física da operadora Claro S.A. em Maceió

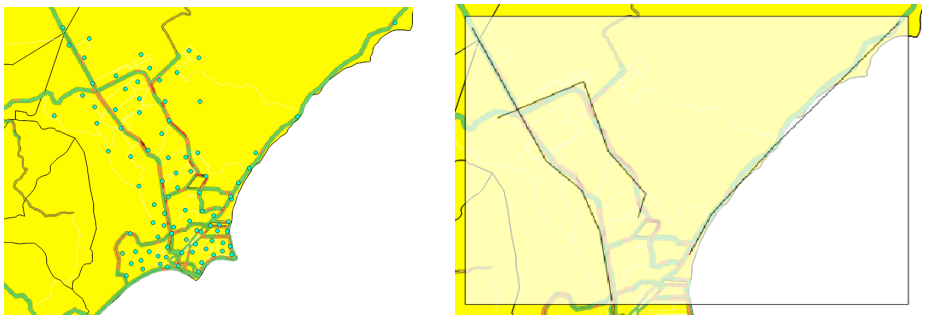
Na Figura 5.17a podemos observar o grafo físico gerado no MATLAB. Através da Figura 5.17b podemos ver a disposição, no mapa, do grafo físico gerado. Em ambas as sub figuras da Figura 5.17, o único gateway encontra-se assinalada através de um quadrado branco delineado a preto, os 20 agregadores através de um circulo delineado com uma cor e as 92 antenas através de um asterisco com

a cor do respetivo agregador. Ainda nas figuras, as ligações a preto formam uma árvore dos caminhos mais curtos para o gateway e as vermelhas são as ligações acrescentadas como "atalhos".

O grafo virtual é gerado de forma análoga à que explicamos na Secção 5.2. Desta forma, esta rede terá 185 vértices dos quais 92 representam antenas, outras 92 representam os vértices intermédios e 1 representa o gateway. Podemos então visualizar o grafo virtual gerado na Figura 5.18.

Fig. 5.18 Rede virtual da operadora Claro S.A. em Maceió

Também neste caso realista, para testarmos os casos parado e movimento, explicados na Subsecção 5.2, necessitamos de simular estradas. As principais estradas que passam por Maceió[2], e que por isso são certamente zonas de grande mobilidade, estão visíveis na Figura 5.19a. Como não conseguimos extrair as coordenadas de nenhuma das estradas para podermos fazer uma simulação correta, escolhemos três e tentamos gerar-las o mais próximo possível da realidade através de linhas poligonais. Podemos observar a simulação das estradas na Figura 5.19b e são as que vamos utilizar para testar o problema.



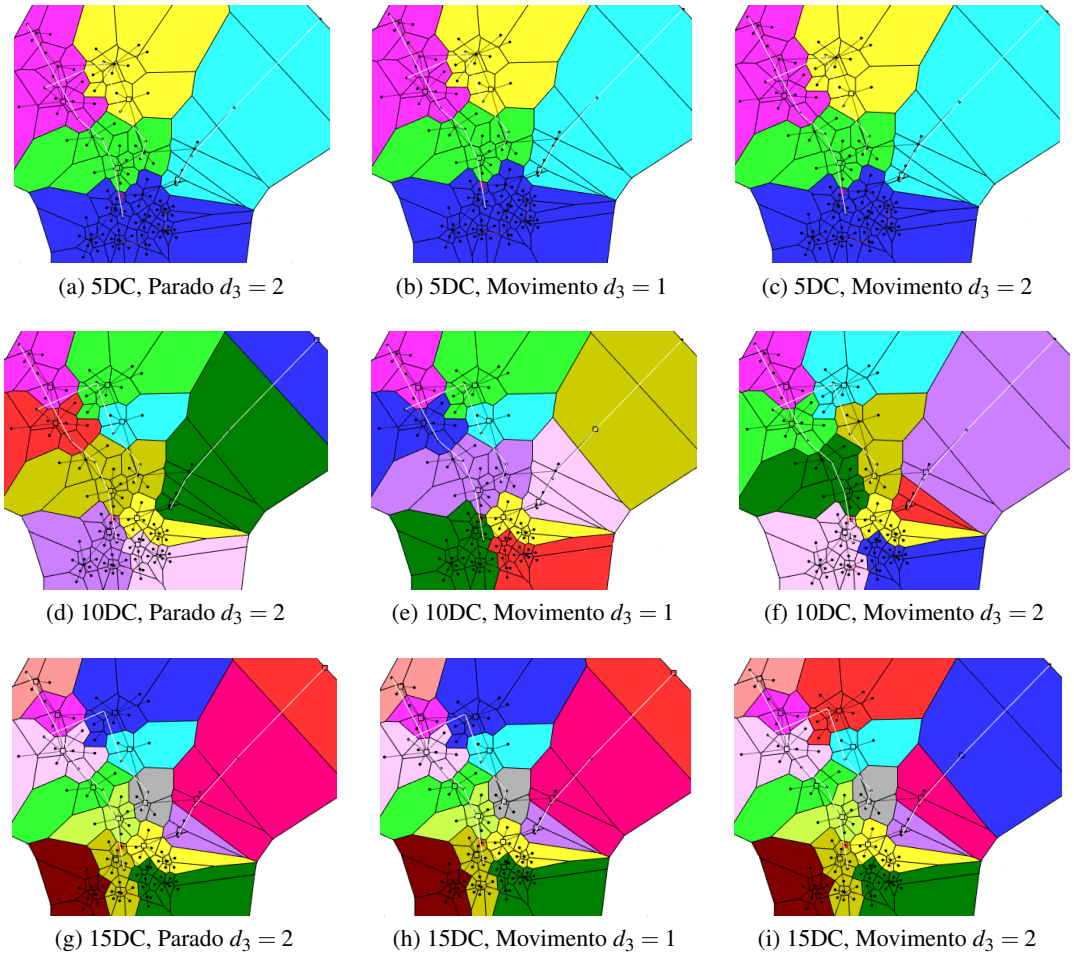
(a) Estradas principais (b) Simulação de uma estrada

Fig. 5.19 Rede de estradas em Maceió

Nos testes que aqui vamos realizar, vamos continuar a considerar capacidades elevadas. Relativamente aos pesos, serão determinados consoante estivermos a considerar o caso parado ou movimento. Posto isto, estamos em condições de recorrer ao IBM ILOG CPLEX Otimization Studio para obtermos

[2]http://www.maceio.al.gov.br/wp-content/uploads/admin/pdf/2015/10/RT-VLT-00-2A0-001_0.pdf nesta ligação, podemos visualizar todas as estradas de maceio sendo que, no nosso caso, apenas considerámos as principais

a solução do problema. Aqui, vamos voltar a minimizar a função f_5 com $d_1 = 3$, $d_2 = 1$. Novamente o parâmetro d_3 pode tomar os valores 1 ou 2. Depois de correrrmos o programa CPLEX, voltamos ao MATLAB para conseguirmos visualizar a solução. As 9 sub figuras da Figura 5.20 são as soluções obtidas para cada situação considerada.



(a) 5DC, Parado $d_3 = 2$ (b) 5DC, Movimento $d_3 = 1$ (c) 5DC, Movimento $d_3 = 2$

(d) 10DC, Parado $d_3 = 2$ (e) 10DC, Movimento $d_3 = 1$ (f) 10DC, Movimento $d_3 = 2$

(g) 15DC, Parado $d_3 = 2$ (h) 15DC, Movimento $d_3 = 1$ (i) 15DC, Movimento $d_3 = 2$

Fig. 5.20 Testes com mobilidade e estradas simuladas de Maceió

Quando consideramos o parâmetro d_3 igual a 2 pretendemos que o problema dê mais importância à minimização da distância da antena ao data center que lhe está associado. Assim, considerar o parâmetro d_3 igual a 1 significa que não estamos tão importados assim quanto à distância referida anteriormente. Desta forma, a diferença entre testar o problema para o caso movimento, considerando o parâmetro d_3 igual a 2 ou igual a 1 é que no primeiro as antenas vão estar associadas a data center mais próximos do que no segundo. Tal pode ser observado comparando a coluna do meio da Figura 5.20 com a coluna da direita da mesma. Sendo que, quando considerámos 5 data centers, é normal que não se observem diferenças porque são poucos face ao número de antenas e agregadores considerados. Comparando as imagens da coluna da esquerda com as da coluna da direita, é na zona por onde passa a estrada mais à direita, ou seja, na estrada mais a este do Brasil onde se verificam as principais diferenças entre as soluções.

5.4 Trabalho futuro

O trabalho aqui desenvolvido foi parcialmente realizado no âmbito do projeto MobiWise: From mobile sensing to mobility advising (P2020 SAICTPAC / 0011/2015), co-financiado pelo COMPETE 2020, Portugal 2020 - Programa Operacional de Competitividade e Internacionalização (POCI), do ERDF (Fundo Europeu de Desenvolvimento Regional) da União Europeia e da Fundação portuguesa para a Ciência e Tecnologia (FCT).

Como este projeto vai prolongar-se por mais alguns meses, vamos dar continuidade ao trabalho que já desenvolvemos e que aqui apresentamos. Assim, o nosso principal objetivo será continuar a trabalhar neste caso realista apresentado na Secção 5.3. Nele vamos tentar considerar as capacidades e outros parâmetros inerentes às redes de comunicações móveis. Isto passa por determinar parâmetros realistas, adaptados a casos reais de utilização e polir o modelo.

Outras aplicações do modelo base de homomorfismo de grafos neste contexto poderiam ser explorados. Como por exemplo, a adaptação do conteúdo dos data centers ao perfil dos utilizadores locais.

Bibliografia

[1] Alleg, A., Ahmed, T., Mosbah, M., Riggio, R., and Boutaba, R. (2017). Delay-aware VNF Placement and Chaining based on a Flexible Resource Allocation Approach. *2017 13th International Conference on Network and Service Management (CNSM)*, IEEE.

[2] Barabási, A.-L. and Albert, R. (1999). Emergence of Scaling in Random Networks. *Science*, 286(5439).

[3] Brewster, R. (2006). Graph Homomorphism Tutorial. Apresentado em *Field's Institute Covering Arrays Workshop*,[http://www.site.uottawa.ca/∼lucia/CA06/TutorialBrewster.pdf].

[4] Carpio, F., Dhahri, S., and Jukan, A. (2017). VNF Placement with Replication for Loac Balancing in NFV Networks. *2017 IEEE International Conference on Communications (ICC)*, IEEE.

[5] Chochlidakis, G. and Friderikos, V. (2016). Mobility Aware Virtual Network Embedding. *IEEE Transactions on Mobile Computing*, 16(5).

[6] Clímaco, J. N., Antunes, C. H. and Alves, M. J. G. (2003). Investigação operacional, Programação linear, Programação linear multicritério, Programação linear multiobjectivo, Sistemas de apoio à decisão *Imprensa da Universidade de Coimbra*, Coimbra.

[7] Fan, W., Li, J., Ma, S., Wang, H., and Wu, Y. (2010). Graph Homomorphism Revisited for Graph Matching. *Proceedings of the VLDB Endowment*, 3(1-2).

[8] Fischer, A., Botero, J. F., Beck, M. T., De Meer, H., and Hesselbach, X. (2013). Virtual Network Embedding: A Survey. *IEEE Communications Surveys & Tutorials*, 15(4).

[9] Luizelli, M. C., Bays, L. R., Buriol, L. S., Barcellos, M. P., and Gaspary, L. P. (2015). Piecing Together the NFV Provisioning Puzzle: Efficient Placement and Chaining of Virtual Network Functions. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE.

[10] Mertens, S. (2006). The easiest hard problem: Number partitioning. *Computational Complexity and Statistical Physics*, 125(2).

[11] Nascimento, M. C. V. (2010). *Metaheurísticas para o problema de agrupamento de dados em grafo*. Tese de Doutorado, Universidade de São Paulo.

[12] Pereira, J. S. (2009). Matemática discreta: Grafos, redes, aplicações. *Luz da Vida, Coimbra, Portugal*.

[13] Rost, M. and Schmid, S. (2018). NP-Completeness and Inapproximability of the Virtual Network Embedding Problem and Its Variants. *arXiv preprint arXiv:1801.03162*.

[14] Sato, C. M. (2008). Homomorfismos de grafos. Tese de Mestrado, Universidade de São Paulo.

Anexo A

Lista de programas

Para ter acesso aos códigos na íntegra, contactar a autora ou consultar a versão online.

Lista de programas MATLAB

Lista de programas de geração

• O código do ficheiro GrafoDeChegada.m gera o grafo de chegada (Barabási-Albert) da Secção 3.1.

Função GrafoDeChegada(NumP,capNump,capAresta,no,m)
Input: Nump - número de vértices do grafo; capNump - capacidade dos vértices; capAresta - capacidade das arestas; no - tamanho da semente; m - grau médio (Usar m<=no).
Cria a matriz de adjacência do grafo
Cria a visualização do grafo a partir da matriz de adjacência
Cria o ficheiro .dat com a informação do grafo gerado para utilizar no CPLEX
Cria o ficheiro .txt com a informação do grafo gerado para utilizar no MATLAB

• O código do ficheiro GrafoOrientadoDePartida.m gera o grafo orientado de partida (Cadeias Disjuntas) da Secção 3.2.

Função GrafoOrientadoDePartida(NumV,NumRep,pesosNumV,pesosAresta)
Input: NumV - número de vértices de cada cadeia; NumRep- número de repetições da cadeia; pesosNumV - pesos dos vértices; pesosAresta - pesos das arestas.
Cria a matriz que identifica a que cadeia pertencem os vértices e arestas
Cria a matriz de adjacência do grafo
Cria o vetor dos pesos dos vértices
Cria a visualização do grafo a partir da matriz de adjacência
Cria o ficheiro .dat com a informação do grafo gerado para utilizar no CPLEX
Cria o ficheiro .txt com a informação do grafo gerado para utilizar no MATLAB

• O código do ficheiro GrafoDeChegada_Barabasi.m gera o grafo de chegada (Barabási-Albert) da Secção 3.3.

Função GrafoDeChegada_Barabasi(NumP,no,m)
Input: NumP - número de vértices do grafo; no - tamanho da semente; m - grau médio (Usar m<=no).
Cria a matriz de adjacência do grafo
Cria a visualização do grafo a partir da matriz de adjacência
Cria o ficheiro .txt com a informação do grafo gerado para utilizar no MATLAB

• O código do ficheiro GrafoOrientadoDePartida_Barabasi.m gera o grafo orientado de partida (Barabási-Albert) da Secção 3.3.

Função GrafoOrientadoDePartida_Barabasi(NumV,no,m)
Input: NumV - número de vértices do grafo; no - tamanho da semente; m - grau médio (Usar m<=no).
Cria a matriz de adjacência do grafo
Cria a visualização do grafo a partir da matriz de adjacência
Cria o ficheiro .txt com a informação do grafo gerado para utilizar no MATLAB

• O código do ficheiro Dados.m gera os dados do grafo de chegada (Barabási-Albert) e do grafo orientado de partida (Barabási-Albert) da Secção 3.3.

Função Dados(capNump,capArestap,pesosNumV,pesosArestaV)
Input: capNump - capacidade dos nós de chegada; capArestap - capacidade das arestas de chegada; pesosNumV - pesos dos vértices de partida; pesosArestaV - pesos das arestas de partida.
Lê os dados guardados do grafo de chegada
Lê os dados guardados do grafo orientado de partida
Cria o ficheiro .dat com a informação do grafo gerado para utilizar no CPLEX

• O código do ficheiro GrafoFisico.m gera o grafo físico da Secção 4.3.

Função GrafoFisico(n,m,CapacidadeDC,CapacidadeArestaP)
Input: n x m número de vértices do grafo físico; CapacidadeDC - Capacidade dos vértices que são data centers; CapacidadeArestaP - Capacidade das arestas do grafo.
Cria os vetores X e Y com as coordenadas de cada um dos pontos
Através da função delaunay identifica os vizinhos de cada ponto e guardar a informação na matriz vizinhos
Todo o ponto i é vizinho do própio i e acrescenta informação à matriz vizinhos
Cálculo da matriz das distancias reais entre todos os pontos
Cria os vetores z, w e distancia para criar o grafo.
Definir o gateway como um ponto mais central do grafo
Cria a matriz de adjacência do grafo
Altera o grafo para o grafo da arvore dos caminhos mais curtos para o gateway

Acrescentar ligações no grafo em função da distância.
Cria a visualização do grafo físico com voronoi
Cria o ficheiro .dat com a informação do grafo gerado para utilizar no CPLEX
Cria ficheiros .txt com a informação do grafo gerado para utilizar no MATLAB

• O código do ficheiro GrafoVirtual.m gera o grafo virtual da Secção 4.3.

Função GrafoVirtual(PesoDC, PesoArestasV)
Input: PesoDC - peso dos vertices que são data centers; PesoArestasV - peso das arestas do grafo.
Lê os dados do grafo físico para gerar o grafo virtual
Cria a matriz de adjacência do grafo
Cria a visualização do grafo virtual
Cria o vetor dos pesos dos vértices
Cria o ficheiro .dat com a informação do grafo gerado para utilizar no CPLEX

• O código do ficheiro GrafoFisicoEVirtualMobilidade.m gera o grafo físico e o grafo virtual com mobilidade da Secção 5.2.

Função GrafoFisicoEVirtualMobilidade(n,m,CapacidadeDC,CapacidadeArestaP,PesoDC)
Input: n x m número de vértices do grafo físico; CapacidadeDC - Capacidade dos vertices que são data centers; CapacidadeArestaP - Capacidade das arestas do grafo físico; PesoDC - Peso dos vértices que são data center.
Criar os vetores X e Y com as coordenadas de cada um dos pontos
Através da função delaunay identifica os vizinhos de cada ponto e guardar a informação na matriz vizinhos
Todo o ponto i é vizinho do próprio i e acrescenta informação à matriz vizinhos
Cálculo da matriz das distancias reais entre todos os pontos
Cria os vetores z, w e distancia para criar o grafo
Definir o gateway como um ponto mais central do grafo
Cria a matriz de adjacência do grafo físico com mobilidade
Altera o grafo para o grafo da arvore dos caminhos mais curtos para o gateway
Acrescentar ligações no grafo em função da distância
Cria a visualização do grafo físico com mobilidade com voronoi
Cria o ficheiro .dat com a informação do grafo físico com mobilidade gerado para utilizar no CPLEX
Determina o valor inicial dos pesos das arestas do grafo virtual com mobilidade
Cria o ficheiro .txt com a informação dos pesos das arestas do grafo virtual com mobilidade para utilizar no MATLAB
Cria a matriz de adjacência do grafo virtual com mobilidade
Cria a visualização do grafo virtual com mobilidade
Cria o vetor dos pesos dos vértices
Cria o ficheiro .txt com a informação do grafo virtual com mobilidade gerado para utilizar no MATLAB

Cria o ficheiro .dat com a informação do grafo virtual com mobilidade gerado para utilizar no CPLEX

• O código do ficheiro GerarMobilidadeParado.m gera os novos pesos para o grafo virtual com mobilidade considerando o caso Parado da Secção 5.2.

Função GerarMobilidadeParado
Lê os dados dos grafos gerados com mobilidade
Simulação da estrada para a mobilidade
Cria um vetor com o número das antenas por onde passa a estrada
Localiza a estrada na visualização do grafo
Define o valor a somar ao peso se a estrada passar pela antena
Alteração dos pesos das arestas
Cria ficheiro .txt com informação os dados da estrada para o MATLAB
Cria ficheiro .txt com informação os dados dos novos pesos para o MATLAB
Cria a matriz de adjacência do grafo virtual
Cria a visualização do grafo virtual com mobilidade
Cria o vetor dos pesos dos vértices
Cria o ficheiro .txt com informação dos dados do grafo virtual com mobilidade para o MATLAB
Cria o ficheiro .dat com a informação do grafo virtual com mobilidade gerado para utilizar no CPLEX

• O código do ficheiro GerarMobilidadeMovimento.m gera os novos pesos para o grafo virtual com mobilidade considerando o caso Mobilidade da Secção 5.2.

Função GerarMobilidadeMovimento
Na sua essência, o código é igual ao GerarMobilidadeParado.m. Apenas as alterações do valor dos pesos são feitos segundo o caso Movimento

• O código do ficheiro GerarMobilidadeRealista.m gera o grafo físico e grafo virtual com mobilidade no caso realista da secção 5.3.

Função GerarMobilidadeRealista(k,CapacidadeDC, CapacidadeArestaP,PesoDC)
Input: k - número de agregadores; CapacidadeDC - Capacidade dos vértices que são data centers; CapacidadeArestaP - Capacidade das arestas do grafo físico; PesoDC - Peso dos vértices que são data center
Importa as coordenadas da localização física das antenas da base de dados
agrupamento das médias-k
Cria as cores para a visualização dos grafos
Determina a localização do gateway
Acrescenta o ponto gateway ao vetor do conjunto de pontos dos agregadores
Cria uma matriz para guardar a informação sobre os agregadores vizinhos

Através da função delaunay identifica os vizinhos de cada agregador e guardar a informação na matriz vizinhos agregadores

Todo o agregador i é vizinho do agregador i e acrescenta informação a vizinhos agregadores

Cria os vetores z, w e distancia para criar o grafo

Cálculo da matriz das distancias reais dos agregadores entre si e entre o gateway.

Cria o grafo com todas as ligações possíveis entre o gateway e os agregadores

Altera o grafo para o grafo da árvore dos caminhos mais curtos dos agregadores para o gateway

Acrescenta ligações no grafo em função da distância entre agregadores

Cria a visualização do grafo físico com mobilidade realista com coloração

Cria o ficheiro .dat com a informação do grafo físico com mobilidade realista gerado para utilizar no CPLEX

Criar uma matriz para guardar a informação sobre as antenas vizinhas

Através da função delaunay identifica as antenas vizinhas e guardar a informação na matriz vizinhos

Toda a antena i é vizinha da própria antena i e acrescenta informação a vizinhos

Determina o valor inicial dos pesos das arestas do grafo virtual com mobilidade realista

Cria o ficheiro .txt com a informação dos pesos das arestas do grafo virtual para utilizar no MATLAB

Cria a matriz de adjacência do grafo virtual com mobilidade no caso realista

Cria a visualização do grafo virtual com mobilidade realista

Cria o vetor dos pesos dos vértices

Cria o ficheiro .txt com a informação do grafo virtual com mobilidade caso realista gerado para utilizar no MATLAB

Cria o ficheiro .dat com a informação do grafo virtual com mobilidade caso realista gerado para utilizar no CPLEX

• O código do ficheiro GerarMobilidadeParadoRealista.m gera os novos pesos para o grafo virtual com mobilidade considerando o caso Mobilidade da Secção 5.3.

Função GerarMobilidadeParadoRealista

Lê os dados dos grafos com mobilidade realista gerados

Simulação da estrada para a mobilidade

Cria vetores com o número das antenas por onde passa a estrada

Localiza a estrada na visualização do grafo

Define o valor a somar ao peso se a estrada passar pela antena

Alteração dos pesos das arestas

Cria ficheiro .txt com a informação dos dados da estrada para o MATLAB

Cria ficheiro .txt com a informação dos dados dos novos pesos para o MATLAB

Cria a matriz de adjacência do grafo virtual com mobilidade caso realista

Cria a visualização do grafo virtual com mobilidade

Cria o vetor dos pesos dos vértices

Cria o ficheiro .txt com informação dos dados do grafo virtual com mobilidade para o MATLAB

Cria o ficheiro .dat com a informação do grafo virtual com mobilidade gerado para utilizar no CPLEX

• O código do ficheiro GerarMobilidadeMovimentoRealista.m gera os novos pesos para o grafo virtual com mobilidade considerando o caso Mobilidade da Secção 5.3.

Função GerarMobilidadeMovimentoRealista

Na sua essência, o código é igual ao GerarMobilidadeParadoRealista.m. Apenas as alterações do valor dos pesos são feitos segundo o caso Movimento

Lista de programas de visualização da solução

• O código do ficheiro SolucaoGC_GOP.m permite-nos visualizar a solução do mapeamento do grafo orientado de partida (Cadeias Disjuntas) para o grafo de chegada (Barabási-Albert) da Secção 3.3.

Função SolucaoGC_GOP

Lê os dados guardados do grafo de chegada

Lê os dados guardados do grafo orientado de partida

Lê os dados guardados da solução

Apresenta o grafo de chegada com os vértices e arestas utilizados no mapeamento a vermelho

• O código do ficheiro SolucaoBarabasi.m permite-nos visualizar a solução do mapeamento do grafo orientado de partida (Barabási-Albert) para o grafo de chegada (Barabási-Albert) da Secção 3.3.

Função SolucaoBarabasi

Na sua essência, o código é igual ao SolucaoGC_GOP.m.

• O código do ficheiro SolucaoGF_GV.m permite-nos visualizar a solução do mapeamento do grafo virtual para o grafo físico da Secção 4.3.

Função SolucaoGF_GV

Lê os dados guardados do grafo físico

Lê os dados guardados da solução

Cria as cores para a visualização dos grafos

Cria a visualização do grafo físico colorido e com o voronoi colorido

• O código do ficheiro solucaoFronteiraPareto.m permite-nos visualizar a fronteira de Pareto da Secção 4.4.

Função solucaoFronteiraPareto

Lê os dados

Apresenta a fronteira de Pareto

• O código do ficheiro SolucaoMobilidade.m permite-nos visualizar a solução do mapeamento do grafo virtual com mobilidade para o grafo físico com mobilidade da Secção 5.2.

Função SolucaoMobilidade

Na sua essência, o código é igual ao SolucaoGF_GV.m.

• O código do ficheiro solucaoFronteiraParetoMob.m permite-nos visualizar a fronteira de Pareto da Secção 5.2.

Função solucaoFronteiraParetoMob

Lê os dados

Apresenta a fronteira de Pareto

• O código do ficheiro SolucaoMobilidadeRealista.m permite-nos visualizar a solução do mapeamento do grafo virtual com mobilidade para o grafo físico com mobilidade ambos do caso realista da Secção 5.3.

Função SolucaoMobilidadeRealista

Lê os dados das antenas

Na sua essência, o restante código é igual ao SolucaoGF_GV.m.

Lista de programas CPLEX

• O código do ficheiro ImplementacaoGCGOP.mod determina a solução do mapeamento do o grafo orientado de partida (Cadeias Disjuntas) para o grafo de chegada (Barabási-Albert) da Secção 3.3.

• O código do ficheiro ImplementacaoBarabasi.mod determina a solução do mapeamento do o grafo orientado de partida (Barabási-Albert) para o grafo de chegada (Barabási-Albert) da Secção 3.3.

• O código do ficheiro ImplementacaoGFGV.mod determina a solução do mapeamento do o grafo virtual para o grafo Físico da Secção 4.3.

• O código do ficheiro ImplementacaoFronteiraPareto.mod determina os valores da fronteira de Pareto da Secção 4.4.

• O código do ficheiro ImplementacaoMobilidade.mod determina a solução do mapeamento do o grafo virtual com mobilidade para o grafo Físico com mobilidade da Secção 5.2.

• O código do ficheiro ImplementacaoFronteiraParetoMob.mod determina os valores da fronteira de Pareto da Secção 5.2.

• O código do ficheiro ImplementacaoRealista.mod determina a solução do mapeamento do o grafo virtual com mobilidade para o grafo Físico com mobilidade ambos para o caso realista da Secção 5.3.

Na sua essência, o código é comum a todos estes ficheiros e é o seguinte:
Parâmetros de entrada
Função de minimização
Restrições
Guarda a solução obtida

Lista de programas R

• O código do ficheiro SelecionarDados.R permite-nos filtrar os dados do ficheiro BaseDados.csv da Secção 5.3.

Lê a biblioteca "geosphere"
Lê toda a base de dados
Filtra os dados por estados, município e operadora
Converte o formato da latitude e longitude
Define o novo ponto de origem
Calcula as novas coordenadas de acordo com o novo ponto de origem
Quadro de dados com as novas coordenadas
Guarda as coordenadas num ficheiro .csv