Marta Margarida Braz Pascoal

ALGORITMOS PARA A ENUMERAÇÃO DOS K TRAJECTOS MAIS CURTOS

Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Matemática
1997

pagina vazia não numerada para mudar de página

Marta Margarida Braz Pascoal

ALGORITMOS PARA A ENUMERAÇÃO DOS K TRAJECTOS MAIS CURTOS

Dissertação submetida para satisfação parcial dos requisitos do programa de Mestrado em Matemática, na área de especialização em Matemática Aplicada.

Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Matemática
1997

pagina vazia não numerada para mudar de página

Dissertação submetida para satisfação parcial dos requisitos do programa de Mestrado em Matemática, na área de especialização em Matemática Aplicada.

pagina vazia não numerada para mudar de página

Ao Prof. Doutor Ernesto de Queirós Vieira Martins desejo expressar o meu profundo reconhecimento por me ter proposto o tema deste trabalho, pela sua orientação, pela confiança, pelo apoio e pela permanente disponibilidade durante a sua realização.

A todos os que, directa ou indirectamente, me apoiaram durante o período de preparação deste trabalho a minha gratidão.

Ao Departamento de Matemática da Universidade de Coimbra e ao projecto PRAXIS XXI da JNICT ("Junta Nacional de Investigação Científica e Tecnológica"), através do CISUC ("Centro de Informática e Sistemas da Universidade de Coimbra"), o meu agradecimento pelo apoio concedido durante este período.

Índice

Nota Introdutória

O problema do Trajecto Mais Curto é um clássico de Optimização em Redes que vem merecendo a atenção de muitos investigadores desde meados dos anos 50. O mesmo não tem acontecido com a sua generalização, o problema da Determinação Ordenada dos K ($K > 1$) Trajectos Mais Curtos, muito embora seja vasto o campo de potenciais aplicações práticas. Por exemplo e para além da imediata aplicação na determinação de soluções alternativas, em análise de sensibilidade, na resolução do problema do Trajecto mais Curto com restrições adicionais, como subproblema do problema do Trajecto Mais Curto Multiobjectivos, etc..

Surgido ainda nos anos 50, parece dever-se a Hoffman, [12], o algoritmo mais antigo para determinar os K trajectos mais curtos. Alguns anos mais tarde, Dreyfus propôs um algoritmo que determina ordenadamente não os K trajectos entre um dado par de nós mas os K trajectos de um dado nó para todos os restantes, [8]. Este algoritmo baseia-se no facto do $k^{\underline{\text{ésimo}}}$ trajecto mais curto ser constituído por $j^{\underline{\text{ésimos}}}$ trajectos mais curtos, com $j \leq k$ e pode ser considerado como um melhoramento do algoritmo proposto por Hoffman.

Já no início dos anos 70, Yen propôs um algoritmo para determinar os K caminhos (trajectos sem ciclos) mais curtos entre um dado par de nós, [26, 27], algoritmo esse tido como o mais eficiente para este problema até aos nossos dias. De notar que a determinação dos K caminhos é um problema de maior dificuldade, o que será devido ao facto de poder existir um $k^{\underline{\text{ésimo}}}$ caminho que não seja constituído apenas por $j^{\underline{\text{ésimos}}}$ caminhos mais curtos, com $j \leq k$.

Nos fins da década de 70, Shier propôs algoritmos para a determinação ordenada de trajectos, [22, 23, 24], que são generalizações de algoritmos para o problema do trajecto mais curto, conhecidos por algoritmos dos rótulos. Os algoritmos propostos por Shier têm também por suporte a propriedade atrás enunciada.

Mais recentemente, o problema dos K Trajectos Mais Curtos foi abordado por Martins, [14], que propôs um algoritmo baseado no "apagamento" de trajectos. Este algoritmo tem sofrido sucessivos melhoramentos, tendo em vista a diminuição do espaço de memória necessário ao seu funcionamento. Curiosamente, tais melhoramentos resultaram também numa diminuição significativa do tempo de resolução, [1, 2, 3]. Num artigo ainda em fase de publicação, Martins e Santos, [17], propõem um novo melhoramento que reduz substancialmente o espaço de memória permitindo a resolução de problemas de grandes dimensões.

Entretanto, o problema dos K caminhos não tem sido abordado, excepto para se proporem diferentes implementações do algoritmo de Yen, [21], e num artigo de

Katoh, Ibaraki e Mine, [13], onde se propõe um algoritmo para redes não orientadas, tentando-se tirar partido desta característica para obter uma maior eficiência.

Também recentemente, Eppstein, [9], propôs um algoritmo em que utiliza uma mudança de variável que permite fazer a ordenação dos trajectos evitando várias operações aritméticas, o que se traduz numa maior eficiência computacional. Este algoritmo é o que apresenta a melhor ordem de complexidade, no que respeita ao tempo de execução.

Este trabalho inicia-se com uma breve abordagem ao problema do Trajecto Mais Curto, não só por ser um subproblema do problema dos K Trajectos Mais Curtos mas também por ser um caso particular deste ($K = 1$).

Após a descrição do problema da Determinação Ordenada de Trajectos, passa-se ao estudo deste problema, classificando-se os algoritmos em dois grupos: algoritmos que se baseiam na construção de uma árvore e algoritmos cujo suporte é o Princípio de Optimalidade. Da análise do primeiro grupo de algoritmos, resultaram não só novos algoritmos mas também concluímos que todos os desta classe são particularizáveis para a determinação ordenada de caminhos, desde que se tenham alguns cuidados. Assim, os algoritmos que se baseiam na construção de uma árvore são apresentados de um modo sistemático, desde o menos eficiente e mais fácil até ao mais eficiente e mais complexo, quase sempre por introdução de melhoramentos no algoritmo apresentado anteriormente. No que respeita aos algoritmos que se baseiam no Princípio de Optimalidade, descrevem-se as várias versões do algoritmo inicialmente proposto por Martins, passando-se de uma versão à seguinte apresentando-se a motivação e o melhoramento que esta originou.

Estuda-se ainda o problema da Determinação Ordenada de Caminhos, dando-se especial ênfase ao modo como a particularização dos algoritmos que se baseiam na construção de uma árvore é obtida em cada caso. De notar que para este problema, apenas eram conhecidos o algoritmo de Yen (válido para qualquer tipo de redes) e o algoritmo de Katoh, Ibaraki e Mine (válido apenas para redes não orientadas). O algoritmo de Yen (para ordenar caminhos) foi generalizado para o problema da determinação ordenada de trajectos e não o contrário. Foi ainda estudado o algoritmo de Katoh, Ibaraki e Mine. Note-se que os algoritmos que se baseiam no Princípio de Optimalidade não podem ser adaptados para a determinação ordenada de caminhos, dado este princípio não ser válido para este problema.

Finalmente, apresentam-se alguns resultados computacionais quer para a ordenação de trajectos quer para a ordenação de caminhos. No primeiro caso, são comparados o algoritmo de Eppstein com o algoritmo de Martins e Santos. A escolha do algoritmo de Eppstein deveu-se ao facto deste ser o que apresenta a melhor ordem de complexidade. Para a ordenação de caminhos, o algoritmo de Martins, Pascoal e Santos foi comparado computacionalmente com o algoritmo de Yen e com o algoritmo de Katoh, Ibaraki e Mine, tendo-se utilizado apenas redes não orientadas.

De realçar o facto da maior parte dos algoritmos que se apresentam serem, tanto quanto sabemos, originais.

1 Conceitos de Teoria de Grafos

Nesta secção introduzimos algumas noções elementares de Teoria de Grafos, que serão utilizadas ao longo deste trabalho.

Um **grafo** é um par $(\mathcal{N}, \mathcal{A})$, onde $\mathcal{N} = \{v_1, \ldots, v_n\}$ é um conjunto finito, cujos elementos denominaremos por **nós** e $\mathcal{A} = \{a_1, \ldots, a_m\}$ é também um conjunto finito, cujos elementos denominaremos por **arcos**.

Para facilitar a representação dos nós e dos arcos de um grafo, geralmente iremos associar a $v_i \in \mathcal{N}$ o número natural i, com $i \in \{1, \ldots, n\}$, e a $a_k \in \mathcal{A}$ o natural k, com $k \in \{1, \ldots, m\}$. Esta identificação facilita a implementação computacional de grafos, servindo como base à sua representação nos programas que foram elaborados.

Cada arco a_k pode ainda ser identificado por um par, ordenado ou não, $[i, j]$ onde i e j são dois nós do grafo. Se $[i, j]$ for um par ordenado será representado por (i, j) e diz-se um **arco orientado**; caso contrário diz-se um **arco não orientado**. O nó i é denominado o **antecessor** do arco (i, j) e j o seu **sucessor**.

O grafo $(\mathcal{N}, \mathcal{A})$ diz-se **não orientado** se $\mathcal{A}$ contiver apenas pares não ordenados, diz-se **orientado** se contiver apenas pares ordenados e diz-se **misto** se contiver pares ordenados e pares não ordenados.

Dado um grafo $(\mathcal{N}, \mathcal{A})$ qualquer, designamos por **grafo associado** de $(\mathcal{N}, \mathcal{A})$ um grafo não orientado, constituído pelo mesmo conjunto de nós e cujos arcos são os arcos de $\mathcal{A}$ não tendo em conta a sua orientação. Se $(\mathcal{N}, \mathcal{A})$ for não orientado o seu grafo associado coincidirá obviamente com $(\mathcal{N}, \mathcal{A})$.

Graficamente um grafo pode ser representado como na figura 1, onde os nós são representados por círculos, enquanto que os arcos orientados são representados por uma seta e os não orientados por um segmento de recta.

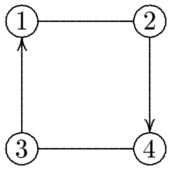

Figura 1

O conceito de grafo que foi apresentado pode ser usado como um modelo matemático representando algumas entidades do quotidiano. Um exemplo simples é o de uma rede de metropolitano ou ferroviária, que pode ser vista como um grafo onde os nós são as estações, e onde existe um arco entre dois nós se existir ligação entre as duas estações respectivas.

A menos que seja dito algo em contrário, iremos considerar no que se segue, que $(\mathcal{N}, \mathcal{A})$ é um grafo orientado e que s e t são dois nós fixos de $(\mathcal{N}, \mathcal{A})$, que denominaremos, respectivamente, por **nó inicial** e **nó terminal**. Iremos considerar ainda, sem perda de generalidade, que existe um único arco entre cada par de nós distintos do grafo, e que sendo $i \in \mathcal{N}$, não existem arcos da forma (i, i) em $(\mathcal{N}, \mathcal{A})$.

Um **trajecto** de s para t em $(\mathcal{N}, \mathcal{A})$ é uma sequência, constituída alternadamente por nós e arcos, da forma $p = \langle v'_1, a'_1, v'_2, \ldots, a'_{\ell-1}, v'_\ell \rangle$, onde $\ell \geq 2$ e:

- $v'_i \in \mathcal{N}$, para $i \in \{1, \ldots, \ell\}$;
- $v'_1 = s$ e $v'_\ell = t$;
- $a'_i \equiv (v'_i, v'_{i+1}) \in \mathcal{A}$, para $i \in \{1, \ldots, \ell - 1\}$.

Convencionamos ainda que para $\ell = 1$, $\langle v'_1 \rangle$ é um trajecto de um nó v'_1 para ele mesmo, isto é, é um trajecto em que não é utilizado nenhum arco.

Caso não haja ambiguidade, um trajecto p poderá ser identificado apenas pelos seus nós.

Sejam i e j nós de $\mathcal{N}$, (i, j) um arco de $\mathcal{A}$ e p um trajecto em $(\mathcal{N}, \mathcal{A})$. Escreveremos $i \in p$ se i for um nó do trajecto p e $(i, j) \in p$ se (i, j) for um arco de p.

Sejam v'_i e v'_j dois nós do trajecto p. Um **subtrajecto** q de p é um trajecto que coincide com p de v'_i até v'_j. Denotaremos q por $\text{sub}_p(v'_i, v'_j)$. É de notar que $\text{sub}_p(v'_k, v'_k) = \langle v'_k \rangle$ para $k \in \{1, \ldots, \ell\}$, e que $p = \text{sub}_p(v'_1, v'_\ell)$.

Consideremos dois trajectos num grafo $(\mathcal{N}, \mathcal{A})$, $p = \langle v^p_1, a^p_1, v^p_2, \ldots, a^p_{\ell_p - 1}, v^p_{\ell_p} \rangle$ e $q = \langle v^q_1, a^q_1, v^q_2, \ldots, a^q_{\ell_q - 1}, v^q_{\ell_q} \rangle$. Se o último nó de p coincide com o primeiro de q, definimos **concatenação** de p e q, que denotamos por $p \diamond q$, como sendo o trajecto

$$p \diamond q = \langle v^p_1, a^p_1, \ldots, v^p_{\ell_p} = v^q_1, a^q_1, \ldots, v^q_{\ell_q} \rangle.$$

Um **caminho** de s para t em $(\mathcal{N}, \mathcal{A})$ é um trajecto com início em s e fim em t, onde:

- $v'_i \neq v'_j$, para $i, j \in \{2, \ldots, \ell - 1\}$ tais que $i \neq j$;
- $s \neq v'_i$ e $t \neq v'_i$, para $i \in \{2, \ldots, \ell - 1\}$.

Um **ciclo** é um caminho de um qualquer nó para ele mesmo. Um caminho pode pois ser visto como um trajecto sem ciclos. Dado um ciclo $\mathcal{C}$ definimos recursivamente $\mathcal{C}^k = \mathcal{C}^{k-1} \diamond \mathcal{C}$, com $\mathcal{C}^1 = \mathcal{C}$, e $k \in \mathbb{N} - \{1\}$. É de notar que $\mathcal{C}^k$ não é um ciclo mas um trajecto constituído pela concatenação de k ciclos $\mathcal{C}$.

A noção de trajecto que foi introduzida para grafos orientados, é extendida naturalmente para grafos não orientados e para grafos mistos. Por exemplo, no grafo da figura 1 a sequência $\langle 1, [1, 2], 2, (2, 4), 4, [4, 3], 3, (3, 1), 1, [1, 2], 2\rangle$ é um trajecto de 1 para 2, que pode ser identificado apenas por $\langle 1, 2, 4, 3, 1, 2\rangle$. O ciclo $\langle 1, 2, 4, 3, 1\rangle$ e o caminho de 1 para 3 $\langle 1, 2, 4, 3\rangle$ são exemplos de subtrajectos daquele trajecto.

Uma **árvore** é um grafo cujo grafo associado não tem ciclos e onde existe um só caminho entre qualquer par de nós. **Árvore com raiz num nó** s, que designaremos simplesmente por árvore, é uma árvore tal que a partir de s existe um caminho para todos os restantes nós.

Denominamos por **ramo** de uma árvore com raiz em s um caminho na árvore desde s até outro nó ao qual só chega ou sai um único arco. Numa árvore com raiz em s, definimos **nível** de um nó i como sendo o número de arcos que existem no caminho de s para i, que se define na árvore. Por conveniência, diz-se que o nível da raiz de uma árvore é zero.

Os conceitos de trajecto, caminho e ciclo, que foram introduzidos, podem ser interpretados, no exemplo de uma rede do metropolitano, como os conceitos que utilizamos usualmente. Um trajecto entre duas estações é uma sequência de estações e ligações entre elas, desde a estação de partida até à estação de chegada. Um trajecto é um caminho se não passarmos mais que uma vez na mesma estação e, por fim, um ciclo é um caminho começando numa estação, em que são utilizadas várias ligações e se volta à estação inicial.

Em geral, numa rede deste tipo poderíamos pensar em descrever algo mais, além de estações e ligações entre elas. Por exemplo, é natural encontrar associados a cada ligação entre estações alguns parâmetros tais como a sua distância, o tempo necessário para ir de uma estação a outra, ou ainda o preço do bilhete. Analogamente, a cada estação podemos, por exemplo, associar o número de bilhetes que nela se vendem por dia, ou mesmo o número de ligações dali para outras estações.

Assim, é natural e por vezes de toda a conveniência, que possamos associar valores numéricos aos nós e/ou arcos de um grafo. A um grafo a cujos nós e/ou arcos se associam valores numéricos denominamos **rede**.

Uma rede será representada simplesmente pelo grafo que lhe está subjacente dizendo-se orientada, não orientada ou mista se este o for.

Consideremos que a cada arco (i, j) duma rede estão associados os valores reais $c_{ij}^1, \ldots, c_{ij}^R$, com $R \geq 1$.

No que se segue e sem perda de generalidade, um arco não orientado $[i, j]$ é substituído por dois arcos orientados, (i, j) e (j, i). Assim, neste trabalho iremos considerar redes mistas e redes não orientadas como sendo redes orientadas e onde cada par $[i, j]$ é um arco da rede se e só se (i, j) e (j, i) forem ambos arcos da rede e tiverem associados os mesmos valores, isto é, $c_{ij}^k = c_{ji}^k$, para $k \in \{1, \ldots, R\}$.

Graficamente uma rede será representada como na figura 2, onde aos arcos do grafo que lhe está associado (i, j) acrescentamos os respectivos valores c_{ij}^k, para $k \in \{1, \ldots, R\}$. Na figura em questão consideramos apenas três valores associados a cada arco, ou seja, $R = 3$.

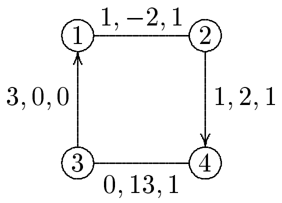

Figura 2

2 Problema do Trajecto Óptimo

Nesta secção vamos descrever um problema clássico da Optimização em Redes, conhecido por problema do Trajecto Óptimo.

São inúmeros os problemas de optimização que podem ser formulados e resolvidos como um problema deste tipo. Além disso, problemas de maior complexidade podem ser resolvidos recorrendo a procedimentos que têm por base a determinação de um trajecto óptimo. Por esta razão, vamos apresentar, ainda que sumariamente, alguns exemplos de problemas do Trajecto Óptimo.

No que se segue, e caso não seja dito nada em contrário, iremos considerar sem perda de generalidade, que $(\mathcal{N}, \mathcal{A})$ é uma rede orientada e que s e t, com $s \neq t$, são, respectivamente, os nós inicial e terminal de qualquer trajecto que se define em $(\mathcal{N}, \mathcal{A})$. Ao arco (i, j) de $(\mathcal{N}, \mathcal{A})$ iremos associar o número real c_{ij}.

Dados dois nós $u, v \in \mathcal{N}$, designaremos por $\mathcal{P}_{uv}$ o conjunto dos trajectos que se definem de u para v em $(\mathcal{N}, \mathcal{A})$ e por $\mathcal{P}$ o conjunto $\mathcal{P}_{st}$. Iremos supor, no que se segue, que existe pelo menos um trajecto em $(\mathcal{N}, \mathcal{A})$ de s para i e pelo menos um trajecto de i para t, onde i é um qualquer nó da rede, isto é, que $\mathcal{P}_{si} \neq \emptyset$ e $\mathcal{P}_{it} \neq \emptyset$, para todo o $i \in \mathcal{N}$.

Seja $c : \mathcal{P}_{uv} \longrightarrow \mathbb{R}$, uma função que atribui, de algum modo, um valor real a cada trajecto definido de u para v em $(\mathcal{N}, \mathcal{A})$.

O problema do Trajecto Óptimo tem por objectivo determinar um trajecto $p^\star \in \mathcal{P}_{uv}$ tal que $c(p^\star)$ seja óptimo em $\mathcal{P}_{uv}$.

Nos problemas do Trajecto Óptimo podemos considerar diversas funções c, e a minimização ou maximização de cada uma dessas funções. Para cada definição de c e da optimização pretendida, teremos um problema do Trajecto Óptimo.

Estes problemas têm várias aplicações práticas, o que é evidenciado pelos nomes pelos quais são conhecidos e que são alusivos às respectivas aplicações.

Iremos descrever em seguida alguns dos problemas do Trajecto Óptimo mais conhecidos, e que podem ser encontrados em [16].

2.1 Problema do Trajecto Mais Curto

Voltando ao exemplo da rede do metropolitano, suponhamos que pretendíamos encontrar uma forma de ir de uma estação para outra percorrendo a menor distância possível (ou pagando o menos possível).

Utilizando $(\mathcal{N}, \mathcal{A})$ para representar a rede em questão, podemos associar a cada arco (i, j) de $(\mathcal{N}, \mathcal{A})$ um valor real c_{ij}, que designaremos por distância (ou custo) do arco (i, j), por analogia com a questão apresentada.

Como é natural, a distância (ou custo) de um trajecto entre duas estações será a soma das distâncias (ou custos) de todas as ligações entre estações intermédias desse trajecto. Assim, dados dois nós $u, v \in \mathcal{N}$, consideremos a função

$$c \ : \ \mathcal{P}_{uv} \longrightarrow \mathrm{I\!R} \ : \ p \ \longrightarrow c(p) = \sum_{(i,j)\in p} c_{ij},$$

denominada também função **distância** ou **custo**.

Por forma a simplificar a notação escreveremos $c(p) = \sum_{p} c_{ij}$.

No problema do Trajecto Mais Curto, dados dois nós u e v de $(\mathcal{N}, \mathcal{A})$, pretendemos pois determinar um trajecto $p^\star$ de u para v, cuja distância (ou custo) seja mínima, isto é, tal que $c(p^\star) \leq c(p)$, para todo o $p \in \mathcal{P}_{uv}$.

De modo análogo poderíamos formular problemas como o de encontrar a forma menos dispendiosa, ou a mais rápida, de ir de uma estação para outra, se c_{ij} representar, respectivamente, o custo ou o tempo necessário para ir de cada nó i a j utilizando o arco (i, j).

Este problema será utilizado em problemas como o dos K Trajectos Mais Curtos, pelo que adiante será apresentada a sua descrição mais pormenorizada, sendo também apresentados, ainda que resumidamente, alguns algoritmos para a sua resolução. Um estudo mais aprofundado pode ser conseguido recorrendo à vasta literatura existente e da qual salientamos [4, 6, 7, 10, 19, 20].

Sem perda de generalidade, quando nos referirmos a este problema iremos considerar c_{ij} como sendo um valor inteiro, apenas com o objectivo de tornar mais simples os programas elaborados.

Se o objectivo pretendido fosse o de maximizar a função $c(p)$ obteríamos o problema do Trajecto Mais Longo, Mais Caro, Mais Lento, etc...

2.2 Problema do Trajecto de Capacidade Máxima

Suponhamos agora, e de acordo com o exemplo anterior, que pretendíamos encontrar uma forma de transportar o maior número possível de pessoas de uma estação para outra, considerando que em cada ligação existe um limite máximo de pessoas a transportar.

Assim, neste problema associamos a cada arco (i, j) de $(\mathcal{N}, \mathcal{A})$ um valor $c_{ij} \in \mathbb{R}^+$, que designamos por capacidade do arco (i, j).

Como é natural, o número de pessoas a transportar, seguindo um dado trajecto entre duas estações, será condicionado pela capacidade das ligações entre as estações desse trajecto. Iremos então considerar a função

$$c \; : \; \mathcal{P}_{uv} \longrightarrow \mathbb{R}^+ \; : \; p \; \longrightarrow c(p) = \min_{(i,j)\in p} c_{ij},$$

que denominaremos por função **capacidade**. Dado $p \in \mathcal{P}_{uv}$, o valor $c(p)$ representa o número de pessoas que é possível transportar da estação u para a estação v, seguindo o trajecto p, isto é, representa a capacidade do trajecto p de u para v.

Por forma a simplificar a notação escreveremos $c(p) = \min_{p} c_{ij}$.

O objectivo deste problema é pois, e como o próprio nome indica, determinar um trajecto $p^\star$ de u para v, cuja capacidade seja máxima, isto é, tal que $c(p^\star) \geq c(p)$, para todo o $p \in \mathcal{P}_{uv}$.

O problema do Trajecto de Capacidade Máxima diz-se do tipo MAXMIN uma vez que pretendemos maximizar uma função definida como sendo o mínimo de um conjunto finito; isto é, queremos obter

$$\max_{p\in\mathcal{P}_{uv}} c(p) = \max_{\mathcal{P}_{uv}} \min_{p} c_{ij}.$$

Analogamente poderíamos considerar um problema do tipo MINMAX. Nesse caso teríamos $c(p) = \max_{p} c_{ij}$ e o objectivo seria encontrar

$$\min_{\mathcal{P}_{uv}} \max_{p} c_{ij}.$$

Este problema é conhecido por problema do Trajecto de Fiabilidade Mínima.

2.3 Problema do Trajecto de Menor Custo por Unidade de Tempo

Voltando ao exemplo da rede do metropolitano, suponhamos que pretendíamos determinar a forma mais barata de ir de uma estação para outra, em termos de custo por unidade de tempo, isto é, com menor custo mas tendo em conta a duração da viagem.

Neste caso será necessário associar a cada arco (i, j) da rede dois valores numéricos, $c_{ij}^1 \in \mathbb{R}$, e $c_{ij}^2 \in \mathbb{R}^+$. O primeiro valor representa o custo do arco e o segundo o tempo que é necessário para o percorrer.

Assim, iremos considerar duas funções c^1 e c^2. A primeira é definida por:

$$c^1 \ : \ \mathcal{P}_{uv} \longrightarrow \mathbb{R} \ : \ p \ \longrightarrow c^1(p) = \sum_p c_{ij}^1$$

e representa o custo de percorrer o trajecto p.

A segunda função é definida por:

$$c^2 \ : \ \mathcal{P}_{uv} \longrightarrow \mathbb{R}^+ \ : \ p \ \longrightarrow c^2(p) = \sum_p c_{ij}^2$$

e representa o tempo necessário para percorrer o trajecto p.

O objectivo deste problema consiste pois em determinar um trajecto $p^\star$ de u para v, que tenha um menor custo por unidade de tempo, isto é, que minimize $c(p) = \dfrac{c^1(p)}{c^2(p)}$ no conjunto $\mathcal{P}_{uv}$.

2.4 Problema do Trajecto de Menor Custo por Unidade de Capacidade

Este problema é análogo ao anterior, embora neste caso se pretenda determinar a forma mais barata de ir de uma estação para outra, em termos de custo por unidade de capacidade.

Neste problema, a cada arco (i, j) da rede são associados dois valores $c_{ij}^1 \in \mathbb{R}$ e $c_{ij}^2 \in \mathbb{R}^+$. Geralmente, designamos o primeiro valor por custo do arco e o segundo por capacidade.

Consideremos então duas funções c^1 e c^2. A função c^1 é definida por:

$$c^1 \ : \ \mathcal{P}_{uv} \longrightarrow \mathbb{R} \ : \ p \ \longrightarrow c^1(p) = \sum_p c_{ij}^1$$

e representa o custo em percorrer o trajecto p; a função c^2 é definida por:

$$c^2 \ : \ \mathcal{P}_{uv} \longrightarrow \mathbb{R}^+ \ : \ p \ \longrightarrow c^2(p) = \min_p c_{ij}^2$$

e representa a capacidade do trajecto p.

O objectivo deste problema é determinar um trajecto $p^\star$ de u para v, que tenha o menor custo por unidade de capacidade, isto é, que minimize $c(p) = \dfrac{c^1(p)}{c^2(p)}$ no conjunto $\mathcal{P}_{uv}$.

Tal como o anterior, este problema por ser interpretado como um problema de Multiobjectivos. Sem pretendermos abordar este tipo de problemas, referimos apenas que é necessário obter uma optimização relativa a duas funções, $c^1(p)$ e $c^2(p)$ (ou mais, se associarmos mais valores a cada arco).

3 Problema do Trajecto Mais Curto

Na secção anterior foi descrito, resumidamente, o problema do Trajecto Mais Curto. Dada a importância deste problema, na presente secção vamos apresentá-lo de forma um pouco mais aprofundada, sendo ainda descritos alguns dos algoritmos conhecidos para a sua resolução.

Relembrando este problema, a cada arco (i,j) associamos o valor c_{ij}, que consideramos ser inteiro, sem perda de generalidade. Pretendemos então minimizar em $\mathcal{P}$ a função custo $c(p) = \sum_p c_{ij}$. É de notar que se p não contiver arcos, isto é, se o trajecto p for constituído apenas por um nó, então convencionamos que $c(p) = 0$.

Seja $\mathcal{C}$ um ciclo de $(\mathcal{N}, \mathcal{A})$, $\mathcal{C}$ diz-se um **ciclo negativo** para o problema do Trajecto Mais Curto se $c(\mathcal{C}) = \sum_{\mathcal{C}} c_{ij} < 0$. Uma vez que estamos a supor que $\mathcal{P} \neq \emptyset$, o problema do Trajecto Mais Curto terá sempre solução óptima. Dizemos que este problema é finito se tem solução óptima de custo finito. É demonstrado no lema 3.1 que o problema do Trajecto Mais Curto é finito se e só se a rede não contiver ciclos negativos.

Lema 3.1 *O problema do Trajecto Mais Curto é finito se e só se não existem ciclos negativos em* $(\mathcal{N}, \mathcal{A})$.

Demonstração: Comecemos por demonstrar que se o problema tem solução óptima de custo finito então não existem ciclos negativos na rede.

Suponhamos então que $c(\mathcal{C}) < 0$, para um dado ciclo $\mathcal{C}$ da rede.

Consideremos um nó v de $\mathcal{C}$. Como foi referido anteriormente, supomos que $\mathcal{P}_{si} \neq \emptyset$ e que $\mathcal{P}_{it} \neq \emptyset$, para todo o nó i de $(\mathcal{N}, \mathcal{A})$; portanto, podemos considerar um trajecto de s para v, que denotaremos por p_1, e outro de v para t, que denotaremos por p_2. Assim, qualquer que seja $k \geq 1$, $q_k = p_1 \diamond \mathcal{C}^k \diamond p_2$ é um trajecto de s para t, e

$$c(q_k) = c(p_1) + c(\mathcal{C}^k) + c(p_2) = c(p_1) + k\, c(\mathcal{C}) + c(p_2),$$

pelo que,

$$\lim_{k\to+\infty} c(q_k) = c(p_1) + c(p_2) + \lim_{k\to+\infty} k\, c(\mathcal{C}).$$

Dado que $c(\mathcal{C}) < 0$, então $\lim_{k\to+\infty} c(q_k) = -\infty$ e o problema não é finito.

Para mostrar a implicação contrária devemos mostrar que se não existem ciclos negativos em $(\mathcal{N}, \mathcal{A})$, então o problema do Trajecto Mais Curto é finito. Como veremos adiante, neste caso os algoritmos de rotulação calculam uma solução óptima finita, isto é, uma solução constituída por um número finito de arcos e de nós. Dado que c_{ij} é finito, qualquer que seja o arco (i, j), fica garantida a existência de uma solução óptima de custo finito. □

No que se segue, iremos supor que as redes consideradas não contêm ciclos negativos, isto é, $c(\mathcal{C}) \geq 0$, para todo o ciclo $\mathcal{C}$ em $(\mathcal{N}, \mathcal{A})$, o que garante a finitude do problema.

Caso não existam ciclos negativos em $(\mathcal{N}, \mathcal{A})$ o problema do Trajecto Mais Curto tem sempre uma solução óptima que é um trajecto que não contém ciclos, ou seja, um caminho. Este resultado é demonstrado no lema 3.2.

Lema 3.2 *Suponhamos que $c(\mathcal{C}) \geq 0$, para todo o ciclo $\mathcal{C}$ em $(\mathcal{N}, \mathcal{A})$. Então existe uma solução óptima do problema do Trajecto Mais Curto de s para t em $(\mathcal{N}, \mathcal{A})$ que é um caminho.*

Demonstração: Dado que $\mathcal{P} \neq \emptyset$, pelo lema 3.1 existe $p^\star \in \mathcal{P}$ que é solução óptima finita do problema em questão. Ou seja, $c(p^\star) \leq c(p)$, para todo o $p \in \mathcal{P}$ e $c(p^\star)$ é finito.

Suponhamos que $p^\star$ não é um caminho, isto é, que $p^\star$ se pode escrever na forma

$$p^\star = p_1 \diamond \mathcal{C}_1 \diamond p_2 \diamond \mathcal{C}_2 \diamond \ldots \diamond p_\ell \diamond \mathcal{C}_\ell \diamond p_{\ell+1},$$

onde $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$ são ciclos de $(\mathcal{N}, \mathcal{A})$ e $p_1, \ldots, p_{\ell+1}$ são caminhos na mesma rede, com $\ell \geq 1$. Então, $c(p^\star) = \sum_{i=1}^{\ell+1} c(p_i) + \sum_{i=1}^{\ell} c(\mathcal{C}_i)$.

Por hipótese não existem ciclos negativos na rede, pelo que $c(\mathcal{C}_i) \geq 0$, para $i \in \{1, \ldots, \ell\}$, e $\sum_{i=1}^{\ell} c(\mathcal{C}_i) \geq 0$, ou seja, $\sum_{i=1}^{\ell+1} c(p_i) \leq c(p^\star)$, e $p_1 \diamond \ldots \diamond p_{\ell+1}$ é um caminho de s para t, tal que

$$c(p_1 \diamond \ldots \diamond p_{\ell+1}) \leq c(p^\star) \leq c(p), \text{ para todo o } p \in \mathcal{P}.$$

Concluímos portanto que $p_1 \diamond \ldots \diamond p_{\ell+1}$ é solução óptima do problema do Trajecto Mais Curto e é um caminho de s para t. □

Um outro problema clássico da Optimização em Redes é o problema do Caminho Mais Curto, em que se pretende encontrar um caminho de custo mínimo. O problema do Caminho Mais Curto tem sempre uma solução óptima finita, uma vez que o domínio da respectiva função objectivo é constituído pelo conjunto dos caminhos de s para t, e um caminho não contém ciclos. Além disso, o número de caminhos entre dois nós distintos de uma rede é sempre finito.

De facto, as redes com maior número de caminhos entre dois nós distintos, supondo que existe um único arco entre dois nós, são aquelas que contêm um arco entre quaisquer dois nós distintos da rede, pelo que o seu número de arcos será $n(n-1)$. Estas redes são denominadas **redes completas**.

O número de caminhos entre dois dados nós de uma rede deste tipo é superior a $(n-2)!$. De facto, dados dois nós numa rede deste tipo, existe 1 caminho com 2 nós, $n-2$ caminhos com 3 nós, $(n-2)(n-3)$ caminhos com 4 nós, ..., e $(n-2)!$ caminhos com os n nós da rede. Assim, o número de caminhos entre dois nós de uma rede completa é dado por $1 + \sum_{i=3}^{n} \prod_{j=2}^{i-1}(n-j)$ e $1 + \sum_{i=3}^{n} \prod_{j=2}^{i-1}(n-j) > (n-2)!$.

O facto de $(\mathcal{N}, \mathcal{A})$ conter ou não ciclos negativos não altera a finitude do problema do Caminho Mais Curto, uma vez que, desde que exista um caminho de s para t, este problema tem sempre solução óptima de custo finito.

Existem vários processos para a resolução do problema do Trajecto Mais Curto. Um deles tem por base a formulação do problema em termos de um programa linear, que pode ser resolvido por algoritmos do tipo simplex, que tiram partido da estrutura especial das matrizes dos coeficientes.[1] No entanto, vários outros algoritmos, baseados no chamado Princípio de Optimalidade, permitem também resolver este problema de modo eficiente. Voltaremos a referir-nos a estes algoritmos, com algum pormenor, mais à frente.

Diz-se que o problema do Trajecto Óptimo verifica o **Princípio de Optimalidade** se qualquer trajecto óptimo é constituído por subtrajectos óptimos.

A não existência de ciclos negativos numa rede é condição necessária e suficiente para que o problema do Trajecto Mais Curto verifique o Princípio de Optimalidade, como é demonstrado no lema 3.3.

Lema 3.3 *Qualquer trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$ é constituído por subtrajectos mais curtos se e somente se não existem ciclos negativos em $(\mathcal{N}, \mathcal{A})$.*

Demonstração: Suponhamos que a rede $(\mathcal{N}, \mathcal{A})$ contém pelo menos um ciclo negativo, $\mathcal{C}$, e seja u um nó de $\mathcal{C}$. Uma vez que $\mathcal{P}_{su} \neq \emptyset$ e $\mathcal{P}_{ut} \neq \emptyset$, podemos considerar dois trajectos p_1 e p_2, de s para u e de u para t, respectivamente.

[1] Dado não terem interesse na resolução do problema dos K Trajectos Mais Curtos, os algoritmos do tipo simplex não são abordados neste trabalho.

O problema do Trajecto Mais Curto de s para t nesta rede não é finito. De facto, seja $q_k = p_1 \diamond \mathcal{C}^k \diamond p_2$, $k \geq 1$, uma sucessão de trajectos de s para t na rede considerada; designando por q_∞ o $\lim_{k\to+\infty} q_k$ e dado que $\lim_{k\to+\infty} c(q_k) = -\infty$, então q_∞ é solução óptima do problema, embora não seja finita. Além disso, o subtrajecto p_1 de q_k não é o mais curto de s para u, para todo o $k \geq 1$, uma vez que $p_1 \diamond \mathcal{C}^k$ é também trajecto de s para u e $c(p_1) > c(p_1 \diamond \mathcal{C}^k)$.

Consideremos agora que a rede $(\mathcal{N}, \mathcal{A})$ não contém ciclo negativos. Então, pelo lema 3.1, o problema do Trajecto Mais Curto é finito. Seja $p^\star \in \mathcal{P}$ um trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$; isto é, $c(p^\star) \leq c(p)$, para todo o $p \in \mathcal{P}$. Quaisquer que sejam u e v, nós distintos de $p^\star$, pretendemos mostrar que $\mathrm{sub}_{p^\star}(u, v)$ é um trajecto mais curto de u para v.

Suponhamos o contrário, isto é, que existe um trajecto q, de u para v, tal que $c(q) < c(\mathrm{sub}_{p^\star}(u, v))$. Então $\mathrm{sub}_{p^\star}(s, u) \diamond\ q \diamond \mathrm{sub}_{p^\star}(v, t)$ é um trajecto de s para t e

$$\begin{aligned} c(\mathrm{sub}_{p^\star}(s,u) \diamond q \diamond \mathrm{sub}_{p^\star}(v,t)) &= c(\mathrm{sub}_{p^\star}(s,u)) + c(q) + c(\mathrm{sub}_{p^\star}(v,t)) \\ &< c(\mathrm{sub}_{p^\star}(s,u)) + c(\mathrm{sub}_{p^\star}(u,v)) + c(\mathrm{sub}_{p^\star}(v,t) \\ &= c(p^\star), \end{aligned}$$

pelo que $p^\star$ não é o trajecto mais curto, o que contraria a hipótese. $\square$

A primeira parte deste lema pode ser exemplificada considerando a rede da figura 3, contendo o ciclo $\mathcal{C} = \langle 1, 2, 1\rangle$ de custo $c(\mathcal{C}) = -1$.

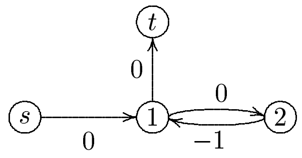

Figura 3

Seja $q_k = \langle s, 1\rangle \diamond \mathcal{C}^k \diamond \langle 1, t\rangle$, com $k \geq 1$, uma sucessão de trajectos de s para 1 nesta rede. A solução óptima deste problema é $\lim_{k\to+\infty} q_k$ e o subtrajecto $\langle s, 1\rangle$ de q_k não é o mais curto de s para 1, para todo o $k \geq 1$, uma vez que $c(\langle s, 1\rangle) > c(\langle s, 1\rangle \diamond \mathcal{C}^k)$.

Tendo por base o Princípio de Optimalidade, podemos definir uma classe de algoritmos designados por **algoritmos de rotulação**. Estes algoritmos determinam não só o trajecto mais curto de s para t, mas também uma árvore com raiz em s e onde o trajecto de s para i, qualquer que seja $i \in \mathcal{N} - \{s\}$, é o mais curto. A esta árvore damos o nome de **árvore dos trajectos mais curtos com raiz em** s. É de notar que esta árvore contém o trajecto pretendido, ou seja, o trajecto mais curto de s para t.

3.1 Algoritmo Geral de Rotulação

Iremos em seguida descrever o algoritmo geral de rotulação que, dada uma rede sem ciclos negativos, determina a árvore dos caminhos mais curtos com raiz em s.

Consideremos o conjunto auxiliar X, constituído pelos nós para os quais já se determinou um trajecto a partir de s e a partir dos quais ainda se poderão determinar melhores alternativas para os trajectos determinados.

A cada nó i da rede associamos um par, (π_i^s, ξ_i^s), que denominamos por **rótulo** de i (daí o nome de algoritmo de rotulação). O valor de π_i^s representará o custo do melhor trajecto de s para i determinado numa dada altura da execução do algoritmo; o valor de ξ_i^s designará o nó que antecede i nesse trajecto. Inicialmente, para $i \neq s$, é atribuído a π_i^s o valor $+\infty$ que é um majorante dos seus possíveis valores. (Sempre que possível é atribuído a π_i^s um valor melhor, ou seja, um valor inferior.)

O algoritmo geral de rotulação é apresentado em seguida no algoritmo 3.1.

Algoritmo 3.1: *Algoritmo geral de rotulação*

$\pi_s^s \longleftarrow 0$;
`Para` (todo o $i \in \mathcal{N} - \{s\}$) `Fazer` $\pi_i^s \longleftarrow +\infty$;
$X \longleftarrow \{s\}$;
`Enquanto` ($X \neq \emptyset$) `Fazer`
    $i \longleftarrow$ um dos nós do conjunto X;
    $X \longleftarrow X - \{i\}$;
    `Para` (todo o $(i,j) \in \mathcal{A}$ tal que $\pi_j^s > \pi_i^s + c_{ij}$) `Fazer`
        $\pi_j^s \longleftarrow \pi_i^s + c_{ij}$;
        $\xi_j^s \longleftarrow i$;
        `Se` ($j \notin X$) `Então` $X \longleftarrow X \cup \{j\}$;
    `FimPara`
`FimEnquanto`

Este algoritmo pode ser implementado computacionalmente de várias formas, dependendo cada uma delas da estrutura de dados utilizada para representar o conjunto X. Essa estrutura determina, em princípio, não só o nó a retirar de X em cada passo, mas também o modo como X é manipulado, o que não é especificado no algoritmo.

Supondo que $c_{ij} \geq 0$ para todos os arcos (i,j) da rede, o algoritmo de Dijkstra, [5, 7], é uma variante do algoritmo de rotulação onde o nó, $i \in \mathcal{N}$, escolhido em X é o de menor custo naquele conjunto. Este algoritmo é descrito resumidamente em 3.2.

Algoritmo 3.2: *Algoritmo de Dijkstra*

$\pi_s^s \longleftarrow 0$;
`Para` (todo o $i \in \mathcal{N} - \{s\}$) `Fazer` $\pi_i^s \longleftarrow +\infty$;
$X \longleftarrow \{s\}$;
`Enquanto` ($X \neq \emptyset$) `Fazer`
  $i \longleftarrow$ nó de X tal que $\pi_i^s \leq \pi_j^s$, qualquer que seja $j \in X$;
  $X \longleftarrow X - \{i\}$;
  `Para` (todo o $(i,j) \in \mathcal{A}$ tal que $\pi_i^s + c_{ij} < \pi_j^s$) `Fazer`
    $\pi_j^s \longleftarrow \pi_i^s + c_{ij}$;
    $\xi_j^s \longleftarrow i$;
    `Se` ($j \notin X$) `Então` $X \longleftarrow X \cup \{j\}$;
  `FimPara`
`FimEnquanto`

Ao contrário do algoritmo geral de rotulação, o algoritmo de Dijkstra é dito de rótulos definitivos. Esta denominação é devida ao facto de a partir do momento em que um nó i é retirado do conjunto X, π_i^s é o valor do trajecto mais curto de s para i, isto é, o rótulo de i não voltará a ser alterado. Este resultado é demonstrado no lema 3.4.

Lema 3.4 *Admitamos que $c_{ij} \geq 0$, para todo o arco $(i,j) \in \mathcal{A}$. No algoritmo de Dijkstra, o rótulo de um nó não é melhorado a partir do momento em que este é escolhido no conjunto X.*

Demonstração: Seja v um nó que num passo do algoritmo de Dijkstra foi removido do conjunto X. Suponhamos que existe um nó i, retirado posteriormente de X, a partir do qual é possível rotular novamente v, ou seja, $\pi_i^s + c_{iv} < \pi_v^s$.

Suponhamos que, quando v foi retirado de X, o nó i pertencia a X. Então v foi o nó escolhido com menor rótulo, logo $\pi_i^s \geq \pi_v^s$ e, como por hipótese $c_{ij} \geq 0$ para todos os arcos (i,j) da rede, teríamos $\pi_i^s + c_{iv} \geq \pi_v^s$.

Suponhamos agora que, quando v foi retirado de X, o nó i ainda não tinha pertencido a X. Então i foi rotulado, directa ou indirectamente, a partir de um nó j, pertencente a X naquele momento. Assim, sendo os custos dos arcos da rede não negativos, $\pi_i^s \geq \pi_j^s$ e como $\pi_j^s \geq \pi_v^s$, temos $\pi_i^s \geq \pi_v^s$; logo $\pi_i^s + c_{iv} \geq \pi_v^s$, e portanto v não pode ser rotulado a partir de i.

Concluímos portanto que o rótulo de v é definitivo. □

Sendo um algoritmo de rotulação, este algoritmo determina a árvore dos caminhos mais curtos com raiz em s. Se pretendermos obter apenas o caminho mais curto de s para t podemos interromper o ciclo quando o nó t é rotulado definitivamente, isto é, quando t é retirado de X.

Também este algoritmo pode ser implementado utilizando diferentes estruturas de dados para o conjunto X. Em cada passo do algoritmo pretendemos retirar de X o seu nó com menor rótulo. Assim, é natural pensar, por exemplo, em duas hipóteses: manter o conjunto ordenado ou não ordenado, segundo os valores de π_i^s, com $i \in \mathcal{N}$. O facto de X ser um conjunto ordenado facilita a remoção dos nós em cada passo, tornando-a imediata. No entanto, manter a ordenação de X implica também um maior esforço computacional.

Estudemos em seguida a ordem de complexidade deste algoritmo.

O ciclo `Enquanto` do algoritmo é executado $(n-1)$ vezes, uma vez que basta rotular, ou seja, remover de X, $(n-1)$ nós.

A partir de um nó retirado de X podemos tentar rotular, no máximo, $(n-1)$ novos nós, supondo que existe um arco entre todos os pares de nós distintos. Temos portanto, no máximo, $(n-1)^2$ tentativas de rotulação, pelo que o processo de rotulação é de $\mathcal{O}(n^2)$, no pior dos casos.

Além das tentativas de rotulação temos ainda de considerar as operações efectuadas na pesquisa do nó a retirar em cada passo do ciclo. Admitamos que X não é mantido ordenado, sendo pois necessário efectuar uma pesquisa exaustiva em cada passo do algoritmo, de modo a encontrar o nó de X com menor rótulo.

Supondo que a partir do primeiro nó em X foram colocados $(n-1)$ novos nós, a primeira pesquisa exaustiva realizar-se-ia sobre um conjunto com $(n-1)$ elementos, a segunda sobre um conjunto com $(n-2)$, a $k^{\underline{\text{ésima}}}$ sobre um conjunto com $(n-k)$, e assim sucessivamente até que na última, X teria apenas 1 elemento. Ou seja, o número de comparações efectuadas seria $\sum_{i=1}^{n-1}(n-i) = \frac{1}{2}(n^2-n)$, pelo que o número de comparações efectuadas seria de $\mathcal{O}(n^2)$.

Ou seja, o algoritmo de Dijkstra para determinação do trajecto mais curto de s para t tem, no pior dos casos, uma complexidade de $\mathcal{O}(n^2)+\mathcal{O}(n^2)$, isto é, de $\mathcal{O}(n^2)$.

Um estudo bastante pormenorizado sobre possíveis implementações dos algoritmos de rotulação pode ser visto em [5].

3.2 Árvore dos Trajectos Mais Curtos com Raiz em t

Um problema análogo ao anterior é o problema do trajecto mais curto de todos os nós de $(\mathcal{N}, \mathcal{A})$ para t.

Ambos os algoritmos descritos anteriormente podem determinar, após convenientes alterações, uma árvore com raiz em t e onde o trajecto de todo o nó

$i \in \mathcal{N} - \{t\}$ para t é o mais curto. A esta árvore damos o nome de **árvore dos trajectos mais curtos de todos os nós para** t.

Este problema é análogo ao da determinação da árvore dos trajectos mais curtos de s para todos os nós, com as diferenças de que a raiz da árvore procurada é o nó t e de que pretendemos determinar trajectos de todos os nós para um outro, t. Como no problema anterior, esta árvore contém também o trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$.

Apesar de a adaptação do algoritmo geral de rotulação e do algoritmo de Dijkstra à resolução deste problema ser relativamente simples, parece-nos conveniente a descrição de pelo menos uma delas, dada a sua utilização em problemas apresentados mais adiante.

A cada nó i da rede associamos o par (π_i^t, ξ_i^t), onde o valor de π_i^t representa o custo do melhor trajecto de i para t determinado numa dada altura da execução do algoritmo e o valor de ξ_i^t designa o nó que se segue a i nesse trajecto. Como nos algoritmos anteriores, inicialmente é atribuído a π_t^t o valor 0 e a π_i^t o valor $+\infty$, para $i \in \mathcal{N} - \{t\}$. Sempre que possível esse valor é melhorado.

A adaptação do algoritmo de Dijkstra para determinação da árvore dos trajectos mais curtos de todos os nós para t é apresentada em 3.3.

Algoritmo 3.3: *Algoritmo de Dijkstra, para determinação da árvore dos trajectos mais curtos com raiz em t*

$\pi_t^t \longleftarrow 0$;
`Para` (todo o $i \in \mathcal{N} - \{t\}$) `Fazer` $\pi_i^t \longleftarrow +\infty$;
$X \longleftarrow \{t\}$;
`Enquanto` $(X \neq \emptyset)$ `Fazer`
  $i \longleftarrow$ nó de X tal que $\pi_i^t \leq \pi_j^t$, qualquer que seja $j \in X$;
  $X \longleftarrow X - \{i\}$;
  `Para` (todo o $(j, i) \in \mathcal{A}$ tal que $\pi_i^t + c_{ji} < \pi_j^t$) `Fazer`
    $\pi_j^t \longleftarrow \pi_i^t + c_{ji}$;
    $\xi_j^t \longleftarrow i$;
    `Se` $(j \notin X)$ `Então` $X \longleftarrow X \cup \{j\}$;
  `FimPara`
`FimEnquanto`

4 Problema dos K Trajectos Mais Curtos

Neste parágrafo iremos descrever o problema dos K Trajectos Mais Curtos e alguns dos seus casos particulares. Este problema é uma generalização do problema do Trajecto Mais Curto, descrito anteriormente.

Voltando ao exemplo que temos vindo a utilizar, suponhamos que pretendíamos viajar o mais rapidamente possível entre duas estações, mas que só o poderíamos fazer a partir das 13 horas. A determinação do trajecto mais rápido entre aquelas estações poderia não ser suficiente para resolver o problema; no entanto poderíamos pensar em encontrar, não apenas a via mais rápida, mas as K mais rápidas para ir de uma estação a outra, sendo K um número natural, que pode ser ou não conhecido previamente. Obter-se-ia assim, por ordem não decrescente de custos, uma sequência de K trajectos, na qual poderíamos escolher o que tivesse o horário mais adequado.

No que se segue, iremos considerar que K é um número natural dado e $K > 1$.

No problema dos K Trajectos Mais Curtos pretendemos enumerar K trajectos entre s e t, por ordem não decrescente de custos; isto é, pretendemos determinar um conjunto $\{p_1, \ldots, p_K\} \subseteq \mathcal{P}$ constituído por K trajectos distintos de s para t e tais que:

- p_i é determinado exactamente antes de p_{i+1}, para todo o $i \in \{1, \ldots, K-1\}$;
- $c(p_i) \leq c(p_{i+1})$, qualquer que seja $i \in \{1, \ldots, K-1\}$;
- $c(p_K) \leq c(p)$, para todo o $p \in \mathcal{P} - \{p_1, \ldots, p_K\}$.

Para $K = 1$ este problema reduz-se ao problema do Trajecto Mais Curto.

Dado que neste problema pretendemos determinar os K trajectos mais curtos, são permitidas repetições de nós em cada um dos trajectos, pelo que é possível determinar trajectos com ou sem ciclos. Trata-se pois do problema dos K Trajectos Mais Curtos, dito sem restrições. Podemos no entanto considerar variantes deste problema impondo restrições aos trajectos a determinar. Algumas destas variantes são descritas abreviadamente de seguida.

4.1 Problema dos K Trajectos Disjuntos

Apesar do objectivo deste trabalho não ser o estudo deste problema vamos descrever neste parágrafo, dois subproblemas do problema dos K Trajectos Disjuntos, dado serem exemplos de problemas dos K trajectos com restrições.

Diremos que dois trajectos de s para t são **disjuntos** se não tiverem nós e arcos em comum, excepto obviamente os nós inicial e terminal. Este conceito pode ser extendido trivialmente para mais do que dois trajectos.

De modo a simplificar a notação iremos considerar associados a todo o nó i duma rede $(\mathcal{N}, \mathcal{A})$ os conjuntos $\mathcal{D}(i) = \{j \in \mathcal{N} : (i,j) \in \mathcal{A}\}$, conjunto dos nós sucessores de arcos com início em i, e $\mathcal{R}(i) = \{j \in \mathcal{N} : (j,i) \in \mathcal{A}\}$, conjunto dos nós antecessores de arcos terminando em i.

O objectivo deste problema é determinar os K trajectos disjuntos mais curtos de s para t. Mais pormenores podem ser encontrados em [25].

4.1.1 Problema dos K Trajectos Arco-Disjuntos

De acordo com a definição dada anteriormente, dizemos que dois trajectos de s para t são **arco-disjuntos** se não tiverem arcos em comum.

O objectivo do problema dos K Trajectos Arco-Disjuntos, é determinar os K trajectos mais curtos que não contêm arcos em comum. Este problema pode ser formulado, em termos de um programa linear, na forma:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \\
\text{s.a.} \quad & \sum_{j\in\mathcal{D}(i)} x_{ij} - \sum_{j\in\mathcal{R}(i)} x_{ji} = \begin{cases} K & \text{se} \quad i = s, \\ 0 & \text{se} \quad i \in \mathcal{N} - \{s,t\}, \\ -K & \text{se} \quad i = t, \end{cases} \\
& 0 \leq x_{ij} \leq 1, \ \forall (i,j) \in \mathcal{A}.
\end{aligned}$$

Da solução deste programa linear obtêm-se os K trajectos mais curtos de s para t, arco-disjuntos. É de notar que na solução do programa linear acima, em geral, é necessário identificar ainda cada um dos trajectos.

Estamos assim a assumir que o problema dos K Trajectos Arco-Disjuntos Mais Curtos tem solução. No caso desta não existir, o mesmo acontece ao programa linear acima.

É ainda de realçar que determinar o trajecto mais curto, eliminá-lo da rede e determinar o trajecto mais curto na rede resultante, não resolve o problema. Tal facto pode ser constatado no exemplo apresentado na rede da figura 4, onde o nó inicial é 1 e o terminal é 4.

Nesta rede, os dois trajectos arco-disjuntos mais curtos são $p_1 = \langle 1,2,4\rangle$ e $p_2 = \langle 1,3,4\rangle$. No entanto, $p = \langle 1,2,3,4\rangle$ é o trajecto mais curto de 1 para 4 naquela rede, e nem sequer faz parte da solução óptima do problema.

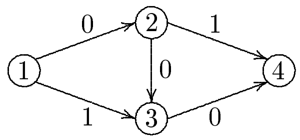

Figura 4

4.1.2 Problema dos K Trajectos Nó-Disjuntos

Ainda de acordo com a definição dada, dizemos que dois trajectos de s para t são **nó-disjuntos** se não tiverem nós em comum, excepto o inicial e o terminal. Todos os trajectos nó-disjuntos são ainda arco-disjuntos, embora o inverso não seja em geral verdade.

O objectivo do Problema dos K Trajectos Nó-Disjuntos é determinar os K trajectos mais curtos de s para t que não contêm nós intermédios em comum.

Este problema pode ser reduzido ao anterior, bastando para isso alterar de forma conveniente a rede $(\mathcal{N}, \mathcal{A})$ dada.

Com base em $(\mathcal{N}, \mathcal{A})$ é construída uma nova rede $(\mathcal{N}^\star,\mathcal{A}^\star)$, como se segue. Todo o nó $i \in \mathcal{N}$ origina dois nós da nova rede, tais que $i', i'' \in \mathcal{N}^\star$ e $(i', i'') \in \mathcal{A}^\star$. Finalmente, todo o arco $(i, j) \in \mathcal{A}$ dá origem ao arco $(i'', j') \in \mathcal{A}^\star$.

É de notar que $\mathcal{A}^\star$ é constituído apenas por arcos da forma (i', i''), para todo o $i \in \mathcal{N}$, e por arcos da forma (i'', j'), para todo o $(i, j) \in \mathcal{A}$. Além disso, os custos associados a cada arco são $c_{i'i''} = 0$, para todo o $i \in \mathcal{N}$, e $c_{i''j'} = c_{ij}$, para $(i, j) \in \mathcal{A}$.

Representando por $\mathcal{D}^\star(i)$ e $\mathcal{R}^\star(i)$ os conjuntos $\mathcal{D}(i)$ e $\mathcal{R}(i)$ na rede $(\mathcal{N}^\star,\mathcal{A}^\star)$, respectivamente, o problema pode então ser formulado, do seguinte modo:

$$\min \sum_{(i,j)\in\mathcal{A}^\star} c_{ij}x_{ij}$$

$$\text{s.a.} \sum_{j\in\mathcal{D}^\star(i)} x_{ij} - \sum_{j\in\mathcal{R}^\star(i)} x_{ji} = \begin{cases} K & \text{se} \quad i = s', \\ 0 & \text{se} \quad i \in \mathcal{N}^\star - \{s', t''\}, \\ -K & \text{se} \quad i = t'', \end{cases}$$

$$x_{ij} \leq 1, \ \forall (i, j) \equiv (x', x'') \in \mathcal{A}^\star,$$

$$x_{ij} \geq 0, \ \forall (i, j) \in \mathcal{A}^\star.$$

É de notar que cada par de trajectos arco-disjuntos em $(\mathcal{N}^\star,\mathcal{A}^\star)$ está associado a um par de trajectos nó-disjuntos em $(\mathcal{N}, \mathcal{A})$.

Também neste caso as soluções não são obtidas ordenadamente, sendo necessário identificar cada trajecto na solução óptima do programa linear.

4.2 Problema dos K Caminhos Mais Curtos

Outro tipo de problema dos K trajectos mais curtos com restrições é o problema dos K Caminhos Mais Curtos, em que se pretende enumerar os K caminhos mais curtos entre dois dados nós de uma rede; isto é, determinar, de forma ordenada, os K caminhos mais curtos entre dois nós de uma dada rede.

Este problema difere do problema dos K Trajectos Mais Curtos por não serem admitidos trajectos com ciclos. Claramente, se a rede não contiver ciclos, todos os trajectos são também caminhos o que implica que os dois problemas sejam equivalentes.

Uma estratégia simples para a resolução deste problema consiste em ir enumerando os trajectos mais curtos e, de entre estes, considerar apenas os que são caminhos. Com base nesta e noutras estratégias é possível considerar vários algoritmos para resolução deste problema.

Mais adiante iremos descrever este problema mais pormenorizadamente, bem como alguns algoritmos para a sua resolução.

5 Enumeração dos K Trajectos Mais Curtos

Na secção anterior foi descrito resumidamente, o problema dos K Trajectos Mais Curtos. Nesta secção iremos apresentar este problema de forma mais aprofundada, descrevendo ainda alguns algoritmos para a sua resolução.

Recordemos que, neste problema, para um dado $K > 1$, pretendemos determinar, por ordem não decrescente de custos, os K trajectos mais curtos entre dois dados nós. Ou seja, pretendemos determinar um conjunto de trajectos $\{p_1, \ldots, p_K\} \subseteq \mathcal{P}$ tais que $c(p_i) \leq c(p_{i+1})$, sendo p_i determinado exactamente antes de p_{i+1}, para todo o $i \in \{1, \ldots, K-1\}$, e $c(p_K) \leq c(p)$, para todo o $p \in \mathcal{P} - \{p_1, \ldots, p_K\}$.

5.1 Árvore dos Trajectos Entre Dois Nós

O conjunto dos trajectos entre dois dados nós de uma rede pode ser organizado numa estrutura que forma uma árvore. Com base nesta estrutura é possível desenvolver algoritmos para a resolução do problema dos K Trajectos Mais Curtos.

Iremos descrever em seguida esta árvore.

Consideremos então uma rede $(\mathcal{N}, \mathcal{A})$ e dois dados nós, s e t, dessa rede. Admitamos ainda que $s \neq t$, e que pretendemos enumerar os K trajectos mais curtos de s para t.

Sejam p e q dois trajectos distintos de s para t em $(\mathcal{N}, \mathcal{A})$; então p e q têm uma parte inicial comum, que é simultaneamente um subtrajecto de p e q, e podendo eventualmente ser constituído apenas pelo nó inicial, s.

Este resultado pode ser generalizado para qualquer número de trajectos.

Seja k um dado número natural tal que $k > 1$, e sejam $p_1, \ldots, p_k$ trajectos distintos de s para t em $(\mathcal{N}, \mathcal{A})$. Para $i \in \{1, \ldots, k\}$, consideremos que p_i é da forma $p_i = \langle v_1^i, a_1^i, v_2^i, \ldots, a_{\ell_i - 1}^i, v_{\ell_i}^i \rangle$. Para não sobrecarregar a notação, representemos por $\mathrm{sub}_i(v_x^i, v_y^i)$ o subtrajecto de p_i desde o nó v_x^i até ao nó v_y^i, com $x, y \in \{1, \ldots, \ell_i\}$. O trajecto p_k coincide com p_i desde s até um seu nó de ordem n_i, ou seja, até $v_{n_i}^k$, para todo o $i \in \{1, \ldots, k-1\}$. O conjunto $\{v_{n_1}^k, \ldots, v_{n_{k-1}}^k\}$ é constituído pelos nós de p_k onde este trajecto se separa de $p_1, \ldots, p_{k-1}$. O nó de $\{v_{n_1}^k, \ldots, v_{n_{k-1}}^k\}$ para o qual n_i é máximo será designado por **nó desvio** de p_k e será denotado por $v_{\alpha_k}^k$. O subtrajecto $\mathrm{sub}_k(s, v_{\alpha_k}^k)$ é o subtrajecto com maior número de nós que p_k tem em comum com algum dos trajectos de $\{p_1, \ldots, p_{k-1}\}$.

Cada trajecto $p_k \in \{p_1, \ldots, p_K\}$ pode pois ser caracterizado por uma parte inicial, um nó desvio e uma parte final. A parte inicial de p_k é $\mathrm{sub}_k(s, v_{\alpha_k}^k)$, o nó desvio é $v_{\alpha_k}^k$ e a parte final, $\mathrm{sub}_k(v_{\alpha_k}^k, t)$. Para o trajecto p_1 consideramos em geral que a parte inicial é constituída apenas por s e que o nó desvio é o nó inicial s. Neste caso a parte final coincide com o próprio trajecto.

Para clarificar, consideremos por exemplo a rede da figura 5, onde os nós inicial e terminal são, respectivamente, $s = 1$ e $t = 5$.

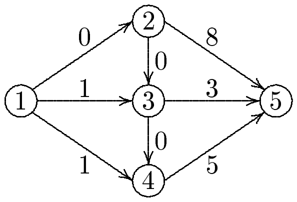

Figura 5

A figura 6 representa o modo como se organizam os trajectos $p_1, \ldots, p_6$ que se definem de s para t na rede da figura 5.

Como podemos observar na figura 6, na representação gráfica daqueles trajectos juntamos a cada nó $i \in \mathcal{N}$ o valor do custo do trajecto de s para i em questão.

É de notar, por exemplo, que o trajecto p_3 se desvia de p_1 no nó $v_3^3 = 3$, e de p_2 no nó $v_1^3 = 1$. Dizemos então que $v_3^3 = 3$ é o nó desvio de p_3 e a sua ordem é $\alpha_3 = 3$.

Podemos então considerar uma árvore formada a partir dos trajectos $p_1, \ldots, p_K$, a que daremos o nome de **árvore dos K trajectos mais curtos para t com raiz em** s.

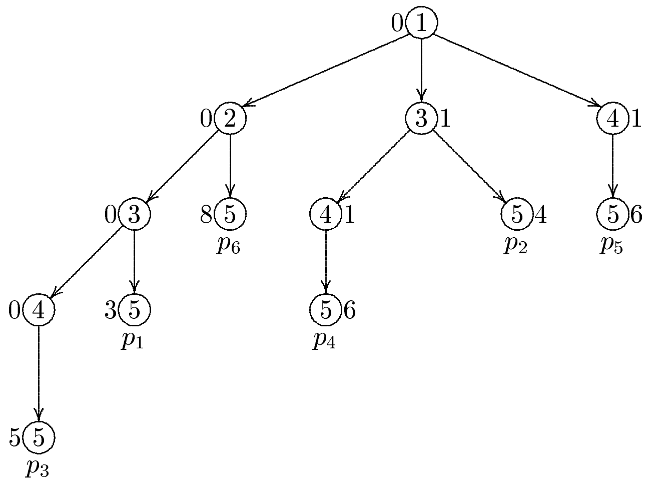

Figura 6

Se $(\mathcal{N}, \mathcal{A})$ não contiver ciclos (como no caso da rede representada na figura 5) todos os trajectos de $(\mathcal{N}, \mathcal{A})$ são caminhos, pelo que podemos considerar um valor qualquer para K, não superior ao número de caminhos de s para t. No entanto, se $(\mathcal{N}, \mathcal{A})$ contiver ciclos, o número de trajectos que podemos definir de s para t não é finito, não sendo pois possível construir a árvore de todos os trajectos para t com raiz em s.

É importante salientar que para assegurar a estrutura de árvore, o mesmo nó da rede deve ser representado de forma diferente quando nela surge mais do que uma vez; isto é, quando pertence a mais do que um trajecto ou quando aparece repetido num mesmo trajecto.

Seja X um conjunto de números naturais representando os nós de uma árvore de trajectos. Como foi referido, é possível (e provável) que um nó da rede faça parte de mais do que um trajecto de s para t. De modo a distinguir os nós da rede que surgem mais do que uma vez nesta árvore, definimos uma função $h : X \to \mathcal{N}$, por forma a que cada nó da árvore seja representado por um número natural em X, todos distintos entre si. A aplicação h indica o nó de $(\mathcal{N}, \mathcal{A})$ que corresponde a um certo nó da árvore.

Utilizando a aplicação h, a árvore dos trajectos de 1 para 5 em $(\mathcal{N}, \mathcal{A})$ pode ser representada como na figura 7, sendo a aplicação h utilizada definida na tabela 1. Poderíamos no entanto ter considerado h definida de modo diferente; desde que associemos diferentes números naturais a cada nó da árvore.

Graficamente é clara a diferença entre o mesmo nó quando este surge em locais diferentes da árvore. Assim, no que se segue a sua representação gráfica será por isso

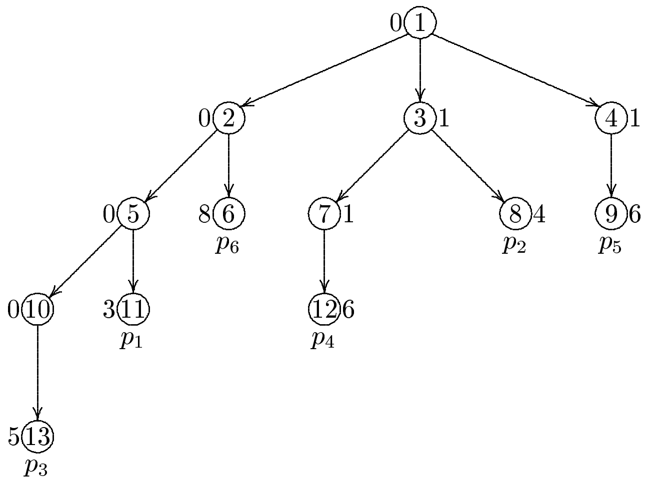

Figura 7

$i \in X$	1	2	3	4	5	6	7	8	9	10	11	12	13
$h(i) \in \mathcal{N}$	1	2	3	4	3	5	4	5	5	4	5	5	5

Tabela 1

realizada apenas com base no número natural que o representa na rede, tal como na figura 6.

Os algoritmos para o problema da enumeração dos K trajectos mais curtos podem ser divididos em dois grupos. O primeiro é constituído pelos algoritmos que se baseiam na forma em árvore como se organizam os trajectos a determinar; estes algoritmos constroem uma sobreárvore da árvore dos K trajectos mais curtos de s para t, isto é, uma árvore que contém a árvore a determinar. O segundo grupo é constituído pelos algoritmos que têm por base o Princípio de Optimalidade, que será enunciado mais adiante, e que é uma generalização do Princípio de Optimalidade para o problema do Trajecto Mais Curto.

Antes de descrever estes algoritmos é de notar que iremos considerar, no que se segue e sem perda de generalidade, que não existem arcos que terminem no nó inicial, s, nem arcos que se iniciem no nó terminal t. Podemos assim considerar também que todos os trajectos de s para t têm, pelo menos, três arcos. De facto, é sempre possível acrescentar dois nós ao conjunto $\mathcal{N}$, S e T, que denominamos respectivamente por **super-nó inicial** e **super-nó terminal**, acrescentando ainda os arcos de custo nulo, (S, s) e (t, T).

Voltando a considerar a rede da figura 5, e após terem sido acrescentados os super-nós inicial e terminal, obtemos a rede da figura 8.

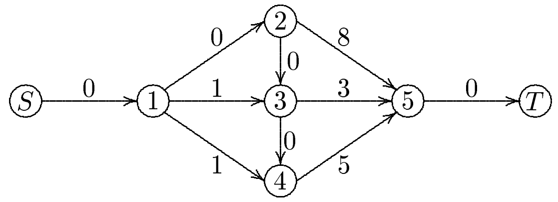

Figura 8

Se p for um trajecto de s para t em $(\mathcal{N}, \mathcal{A})$, então $\langle S, s\rangle \diamond p \diamond \langle t, T\rangle$ é um trajecto de S para T na rede aumentada, exactamente com o mesmo custo de p. É de notar que, como foi referido, este trajecto tem no mínimo três arcos e que, além disso, os novos nós inicial e terminal se encontram nas condições pretendidas.

Descreveremos em seguida os algoritmos destes dois grupos.

5.2 Algoritmos com Suporte na Construção da Árvore dos Trajectos

Como referimos anteriormente, um processo para determinar os trajectos de s para t em $(\mathcal{N}, \mathcal{A})$ é baseado na construção da árvore dos K trajectos mais curtos já descrita, ou na construção de uma árvore que a contenha. Nesta subsecção iremos descrever vários algoritmos baseados nesta construção.

Nos dois primeiros algoritmos a serem apresentados, é realizada praticamente uma pesquisa exaustiva dos trajectos de s para t; isto é, tenta-se ir construindo a árvore de todos os trajectos para t com raiz em s. Esta construção pode ser conseguida de várias formas, como por exemplo, por níveis ou em profundidade; no entanto todas as formas têm em comum o facto de a partir de um nó i da árvore se colocarem todos os nós de $(\mathcal{N}, \mathcal{A})$ que são sucessores de arcos com início em $h(i)$.

No terceiro algoritmo que se apresenta, a árvore é construída a partir da determinação de trajectos e seus desvios. É utilizado um conjunto onde são guardados os trajectos com menor custo que vão sendo calculados.

No quarto algoritmo é utilizado um conjunto análogo. Este algoritmo, embora apresente algumas semelhanças com o terceiro, realiza uma mudança de variável que permite alterar o processo de determinação dos K trajectos pretendidos.

Por fim, o último algoritmo apresentado dentro deste grupo combina, de certa forma, o terceiro e quarto algoritmos. A mudança de variável utilizada no quarto algoritmo é aproveitada para facilitar a determinação dos desvios de trajectos, que é realizada no terceiro algoritmo.

Os cinco algoritmos deste grupo serão estudados de seguida, mais pormenorizadamente.

É de notar que a maior parte destes algoritmos é, tanto quanto temos conhecimento, original, não existindo por isso bibliografia a referenciar. As excepções serão indicadas na altura devida.

5.2.1 Algoritmo de Pesquisa Exaustiva

Como referimos anteriormente, a determinação dos K trajectos mais curtos de s para t passa pela construção da árvore dos trajectos com raiz em s. Neste parágrafo iremos descrever um algoritmo que tenta realizar essa construção dum modo exaustivo, isto é, constrói toda a árvore dos trajectos de s para t em $(\mathcal{N}, \mathcal{A})$.

Os nós da árvore em construção que poderão ainda vir a ser analisados, são guardados num conjunto X', tal que $X' \subseteq X$, sendo posteriormente daí retirados.

Inicialmente, o único elemento em X' é o nó s. Este nó é retirado de X' e a partir dele são acrescentados a X', como novos nós a analisar, todos os sucessores de arcos de $(\mathcal{N}, \mathcal{A})$ com início em s, que passam assim a pertencer à árvore. O procedimento é repetido para os novos nós que ainda não foram analisados.

De um modo genérico, em cada passo do algoritmo é retirado um nó de X', até que este conjunto fique vazio. Se esse nó for o terminal, então foi encontrado um trajecto de s para t em $(\mathcal{N}, \mathcal{A})$.

Como referimos anteriormente, uma vez que estamos a construir uma árvore, de cada vez que o mesmo nó da rede é inserido nesta, essa inserção deverá ser referente a um nó diferente da árvore, pelo que devemos designá-lo de forma diferente.

De modo a distinguir estes casos o algoritmo irá utilizar, para representar cada nó que é colocado na árvore, a respectiva ordem de colocação; isto é, o nó s será o de ordem 1; os nós colocados na árvore a partir de s serão os de ordem 2, ..., e assim sucessivamente. O conjunto X' é constituído pelos valores das ordens dos nós da árvore por analisar.

Para conhecermos o nó da rede ao qual está associada uma certa ordem, definimos uma função $h : X \to \mathcal{N}$, onde $X \subseteq \mathbb{N}$. Na implementação computacional deste algoritmo, h pode ser representada por um vector que na componente $i \in X$ contém o valor de $h(i)$.

Dado um nó na árvore de ordem i, isto é, dado $i \in X$, consideremos que u é o nó da rede que lhe está associado, ou seja, $h(i) = u$, com $u \in \mathcal{N}$. Ao nó i de X associamos o par (π_i^s, ξ_i^s), que designaremos por rótulo de i. Esta designação será por vezes utilizada apenas para referir o valor π_i^s, ao qual ξ_i^s está implicitamente associado. O valor de π_i^s representa o custo do trajecto na árvore de 1 para i, ou seja, de um trajecto em $(\mathcal{N}, \mathcal{A})$ de s para u, e ξ_i^s designa o nó que antecede i nesse trajecto da árvore, ou seja o nó que antecede u no trajecto em questão.

Ao contrário do que acontecia para o problema do Trajecto Mais Curto, neste caso não é necessário melhorar os valores de π_i^s, uma vez que se pretendem guardar todos os trajectos possíveis com início em s. Deste modo não é necessário começar por atribuir a π_i^s um majorante para o valor do trajecto mais curto de s para t, para todo o nó $i \in \mathcal{N} - \{s\}$.

Como referimos, neste algoritmo um novo trajecto para t é identificado sempre que seja retirado de X' um nó i tal que $h(i) = t$. Supomos por isso, e sem perda de generalidade, que o nó t não pode aparecer mais do que uma vez num trajecto. De facto, se existisse um ciclo em $(\mathcal{N}, \mathcal{A})$ contendo t poderíamos acrescentar à rede um super-nó terminal $T \notin \mathcal{N}$ e o arco (t, T) de custo zero. O super-nó terminal seria o nó terminal da rede alterada, não pertencendo a nenhum ciclo, uma vez que não existem arcos com início em T.

Além disso, para que este algoritmo seja finito, a rede $(\mathcal{N}, \mathcal{A})$ não pode conter ciclos. De facto, este algoritmo exige que seja construída toda a árvore dos trajectos de s para t pelo que é finito se e só se o conjunto dos trajectos de s para t o for. No entanto, o conjunto $\mathcal{P}$ é finito se e somente se não existem ciclos na rede, resultado que é enunciado no lema 5.1.

Lema 5.1 *Admitamos que $\mathcal{P}_{si} \neq \emptyset$ e $\mathcal{P}_{it} \neq \emptyset$, para todo o nó $i \in \mathcal{N}$. O conjunto dos trajectos de s para t em $(\mathcal{N}, \mathcal{A})$ é finito se e só se $(\mathcal{N}, \mathcal{A})$ não contiver ciclos.*

Demonstração: Comecemos por demonstrar que se $\mathcal{P}_{st}$ é finito então $(\mathcal{N}, \mathcal{A})$ não contém ciclos.

Suponhamos então que $(\mathcal{N}, \mathcal{A})$ contém pelo menos um ciclo $\mathcal{C}$.

Seja v um dos nós do ciclo $\mathcal{C}$. Como, por hipótese, $\mathcal{P}_{sv} \neq \emptyset$ e $\mathcal{P}_{vt} \neq \emptyset$, existem trajectos p e q de s para v e de v para t, respectivamente. Então $p \diamond q, \ldots, p \diamond \mathcal{C}^k \diamond q, \ldots$, com $k \in \mathbb{N}$, são trajectos distintos de s para t em $(\mathcal{N}, \mathcal{A})$, pelo que o número de trajectos de s para t não é finito.

Para mostrar a implicação contrária devemos mostrar que se não existem ciclos em $(\mathcal{N}, \mathcal{A})$, então $\mathcal{P}$ é finito.

Se a rede não contém ciclos, todo o trajecto entre quaisquer dois nós de $(\mathcal{N}, \mathcal{A})$ é também um caminho. No entanto, como referimos na secção 3, o número de caminhos numa rede é sempre finito, portanto o conjunto $\mathcal{P}$ é também finito. $\square$

Do lema 5.1 concluímos que este algoritmo, quando aplicável, resolve simultaneamente o problema dos K Trajectos Mais Curtos e dos K Caminhos Mais Curtos, uma vez que só é aplicável em redes sem ciclos.

O algoritmo de pesquisa exaustiva é descrito no algoritmo 5.1.

Algoritmo 5.1: *Algoritmo de pesquisa exaustiva*

$ordem \longleftarrow 1$;
$\pi^s_{ordem} \longleftarrow 0$;
$X' \longleftarrow \{ordem\}$;
$h(ordem) \longleftarrow s$;
`Enquanto` $(X' \neq \emptyset)$ `Fazer`
  $i \longleftarrow$ um dos nós do conjunto X';
  $X' \longleftarrow X' - \{i\}$;
  $u \longleftarrow h(i)$;
  `Se` $(u \neq t)$ `Então`
    `Para` (todo o $(u, v) \in \mathcal{A}$) `Fazer`
      $ordem \longleftarrow ordem + 1$;
      $X' \longleftarrow X' \cup \{ordem\}$;
      $h(ordem) \longleftarrow v$;
      $\pi^s_{ordem} \longleftarrow \pi^s_i + c_{uv}$;
      $\xi^s_{ordem} \longleftarrow i$;
    `FimPara`
  `Senão`
    identificar trajecto de 1 para i na árvore;
  `FimSe`
`FimEnquanto`
identificar os K trajectos mais curtos de s para t em $(\mathcal{N}, \mathcal{A})$;

O conjunto X' pode ser manipulado de duas formas:

- Se X' for manipulado como uma lista na forma FIFO (*first in first out*), então X' é um conjunto ordenado de acordo com a posição dos seus elementos. As inserções são realizadas no final do conjunto X' e as remoções no início de X'. Neste caso a construção da árvore é realizada por níveis.

- Se X' for manipulado como uma lista na forma LIFO (*last in first out*), os nós são inseridos e removidos do início de X' e a construção da árvore realiza-se em profundidade.

Com este algoritmo a determinação dos trajectos de s para t em $(\mathcal{N}, \mathcal{A})$ não é realizada de forma ordenada; o que significa que, só no final se podem enumerar os

K trajectos pretendidos. De facto, esta pesquisa de trajectos na árvore é efectuada dependendo da forma como X' é manipulado, e não obrigatoriamente por ordem do custo dos trajectos. No final da execução do algoritmo os vectores h e ξ^s irão conter toda a informação que permite conhecer a árvore dos trajectos, ou seja, todos os trajectos de s para t em $(\mathcal{N}, \mathcal{A})$. No conjunto destes trajectos podem escolher-se os K de menor custo.

Este algoritmo, como qualquer algoritmo de pesquisa exaustiva, é extremamente ineficiente em relação ao número de operações efectuadas bem como ao espaço de memória ocupado. De facto, as operações realizadas neste algoritmo são essencialmente as rotulações dos nós da árvore. Uma vez que é construída a árvore de todos os trajectos para t com raiz em s, o número de rotulações efectuadas é exactamente o número de nós daquela árvore. Para cada nó rotulado é realizada ainda uma comparação; portanto, o número de operações realizadas por este algoritmo é o dobro do número de nós da árvore.

Considerando que $(\mathcal{N}, \mathcal{A})$ é uma rede completa, referimos anteriormente que o número de caminhos que se define de s para t em $(\mathcal{N}, \mathcal{A})$ é da $\mathcal{O}(n!)$. Assim, considerando o pior caso (redes completas), a complexidade do algoritmo, é da $\mathcal{O}(n!)$.

É no entanto de notar que redes completas contêm ciclos, não sendo nesse caso possível aplicar este algoritmo. De facto, como foi referido, o algoritmo de pesquisa exaustiva é válido apenas para **redes acíclicas**, isto é, para redes sem ciclos. No lema 5.2 é demonstrado que numa rede deste tipo com n nós existem, no máximo, 2^{n-2} trajectos entre um dado par de nós. Este caso é atingido quando a rede é **acíclica completa**, isto é, quando $\mathcal{A}$ contém o maior número de arcos possível, embora não contenha ciclos.

Lema 5.2 *Seja $(\mathcal{N}, \mathcal{A})$ uma rede acíclica completa com n nós, com $n \geq 2$. Existem quando muito 2^{n-2} trajectos entre dois dados nós de $(\mathcal{N}, \mathcal{A})$.*

Demonstração: Seja $(\mathcal{N}, \mathcal{A})$ uma rede completa e sem ciclos. É possível ordenar o conjunto $\mathcal{N}$ de modo a que $\mathcal{N} = \{1, \ldots, n\}$ e $(i, j) \in \mathcal{A}$, com $i, j \in \mathcal{N}$, se e só se $i < j$, [11][2]. O pior dos casos é atingido quando $s = 1$ e $t = n$, para $n \geq 2$. Mostremos, por indução sobre n, que existem 2^{n-2} trajectos entre 1 e n em $(\mathcal{N}, \mathcal{A})$. É de notar que estes trajectos são também caminhos uma vez que $(\mathcal{N}, \mathcal{A})$ é acíclica.

Comecemos por considerar que $n = 2$, ou seja, que $(\mathcal{N}, \mathcal{A})$ é constituída apenas por dois nós, $s = 1$ e $t = 2$. Então, $\mathcal{A} = \{(1, 2)\}$ e existe apenas o trajecto $\langle 1, 2 \rangle$ em $(\mathcal{N}, \mathcal{A})$, ou seja, o número de trajectos é $2^0 = 1$.

[2] Pode ser consultado no endereço http://queue.IEOR.Berkeley.EDU/~hochbaum/.

Suponhamos que em redes acíclicas completas com n nós, existem 2^{n-2} trajectos de s para t. Consideremos agora uma rede $(\mathcal{N}, \mathcal{A})$, do mesmo tipo, com $n+1$ nós e mostremos que existem 2^{n-1} trajectos em $(\mathcal{N}, \mathcal{A})$.

Como $(\mathcal{N}, \mathcal{A})$ contém $n+1$ nós, podemos considerar que foi obtida a partir de uma rede com n nós, acrescentando o nó 0 (que passa a ser o inicial) e os arcos $(0, i)$, para todo o $i \in \{1, \ldots, n\}$. Então os trajectos de $s = 0$ para $t = n$ em $(\mathcal{N}, \mathcal{A})$ são da forma $p_i = \langle 0, i \rangle \diamond q_i$, onde $i \in \{1, \ldots, n\}$ e q_i é um trajecto de i para n.

Para $i = 1$, os trajectos q_i, de 1 para n, são trajectos numa rede com n nós e, por hipótese de indução, são 2^{n-2}. Por outro lado, para $i \in \{2, \ldots, n\}$, os trajectos q_i, de i para n são trajectos numa rede acíclica com $n-1$ nós, e portanto p_i são trajectos numa rede acíclica com n nós. Ou seja, novamente por hipótese de indução, existem 2^{n-2} trajectos da forma p_i, com $i > 2$.

Concluímos pois que o número de trajectos em $(\mathcal{N}, \mathcal{A})$, que se definem de 0 para n, é $2 \times 2^{n-2} = 2^{n-1}$, como pretendíamos demonstrar. □

Utilizando um raciocínio semelhante poderíamos demonstrar o lema 5.3.

Lema 5.3 *Seja $(\mathcal{N}, \mathcal{A})$ uma rede acíclica completa com n nós, com $n \geq 2$. A árvore resultante do algoritmo de pesquisa exaustiva para a determinação dos K trajectos mais curtos de s para t em $(\mathcal{N}, \mathcal{A})$ é constituída por 2^{n-1} nós.*

Deste lema podemos concluir de imediato que a complexidade do algoritmo de pesquisa exaustiva é, no pior dos casos, e para redes acíclicas, da $\mathcal{O}(2^n)$

Relativamente ao espaço de memória necessário para a execução deste algoritmo, é preciso guardar a rede e além disso, para cada nó i da árvore, guardar o par (π_i^s, ξ_i^s) e o valor $h(i)$. É necessário ainda utilizar o conjunto X' anteriormente referido, capaz de guardar os nós a colocar na árvore. Este conjunto deverá poder pois conter, todos os nós da árvore, excepto quando muito a sua raiz. O espaço de memória utilizado por este algoritmo, além do utilizado para armazenar a estrutura de rede, é então da mesma ordem que o número de nós da árvore de pesquisa exaustiva, ou seja, da $\mathcal{O}(2^n)$.

5.2.2 Generalização do Algoritmo de Dijkstra

Neste parágrafo iremos descrever um algoritmo para determinação dos K trajectos mais curtos de s para t em $(\mathcal{N}, \mathcal{A})$ que pode ser considerado uma generalização do algoritmo de Dijkstra para o problema do Trajecto Mais Curto.

No que se segue este algoritmo poderá ser designado abreviadamente por algoritmo GD.

De entre as muitas desvantagens do algoritmo de pesquisa exaustiva, podemos referir a necessidade de construir toda a árvore de trajectos de s para t o que, além de ineficiente em termos do tempo utilizado, exige igualmente grande espaço de memória. Poderíamos tentar minimizar esta ineficiência tentando evitar a construção completa da árvore dos trajectos com raiz em s. Isto pode eventualmente ser conseguido se os trajectos forem calculados por ordem não decrescente dos custos.

Analogamente ao que foi feito no algoritmo de Dijkstra para o problema do Trajecto Mais Curto, vamos admitir que $c_{ij} \geq 0$ para todos os arcos (i, j) de $(\mathcal{N}, \mathcal{A})$ e considerar que o elemento a retirar de X' é sempre o elemento de menor rótulo. Como veremos adiante, isto garante que a determinação dos trajectos se efectue ordenadamente, o que, regra geral, evita a determinação de todos os trajectos de s para t em $(\mathcal{N}, \mathcal{A})$.

Este algoritmo apresenta grandes semelhanças com o descrito no parágrafo anterior. O conjunto X' continua a guardar as ordens dos nós por analisar que pertencem à árvore dos trajectos, pelo que ainda é utilizada a função h.

Além disso, como os nós a analisar estão guardados em X', a todo o elemento i de X' associamos um par indicando o custo do trajecto de 1 para i e a ordem do nó que antecede i nesse trajecto.

Como foi referido, em cada novo passo deste algoritmo, e tal como acontecia no algoritmo de Dijkstra, é escolhido o nó em X' com menor custo.

A generalização do algoritmo de Dijkstra é descrita no algoritmo 5.2.

Este algoritmo funciona de modo análogo ao anterior, realizando a construção da árvore dos trajectos de s para t em $(\mathcal{N}, \mathcal{A})$. No entanto, dada a forma como o conjunto X' é manipulado, ou seja, uma vez que é sempre escolhido o nó de X' cujo rótulo tenha menor valor, a construção daquela árvore permite que os trajectos sejam determinados por ordem não decrescente de custos, o que é demonstrado em seguida.

Lema 5.4 *Admita-se que são retirados ℓ ($\ell \geq K$) nós do conjunto X' na generalização do algoritmo de Dijkstra. Então, a sequência $\{\pi_1^s, \ldots, \pi_\ell^s\}$ dos rótulos dos nós retirados de X' é não decrescente.*

Demonstração: Mostremos, por indução sobre $k \in \mathbb{N}$, que $\pi_{k+1}^s \geq \pi_k^s$.

A primeira rotulação realizada é a de s e como $h(1) = s$, $\pi_1^s = 0$. Como π_2^s é o rótulo de um nó da árvore associado ao nó v de $(\mathcal{N}, \mathcal{A})$, colocado a partir de s, então $\pi_2^s = \pi_1^s + c_{sv}$. Uma vez que, por hipótese, todos os arcos de $(\mathcal{N}, \mathcal{A})$ têm custo não negativo, então $\pi_2^s \geq \pi_1^s$.

Suponhamos agora que $\pi_{i+1}^s \geq \pi_i^s$ para $i < k$, ou seja, que os rótulos dos primeiros k nós retirados de X' neste algoritmo foram obtidos por ordem não decrescente.

Algoritmo 5.2: *Generalização do algoritmo de Dijkstra*

$k \longleftarrow 0;$
$ordem \longleftarrow 1;$
$\pi^s_{ordem} \longleftarrow 0;$
$X' \longleftarrow \{ordem\};$
$h(ordem) \longleftarrow s;$
`Enquanto` $(k < K)$ e $(X' \neq \emptyset)$ `Fazer`
  $i \longleftarrow$ nó de X' tal que $\pi^s_i \leq \pi^s_j$, qualquer que seja $j \in X'$;
  $X' \longleftarrow X' - \{i\};$
  $u \longleftarrow h(i);$
  `Se` $(u \neq t)$ `Então`
    `Para` (todo o $(u, v) \in \mathcal{A}$) `Fazer`
      $ordem \longleftarrow ordem + 1;$
      $X' \longleftarrow X' \cup \{ordem\};$
      $h(ordem) \longleftarrow v;$
      $\pi^s_{ordem} \longleftarrow \pi^s_i + c_{uv};$
      $\xi^s_{ordem} \longleftarrow i;$
    `FimPara`
  `Senão`
    $k \longleftarrow k + 1;$
    $p'_k \longleftarrow$ trajecto de 1 para i na árvore;
    $p_k \longleftarrow$ trajecto correspondente a p'_k em $(\mathcal{N}, \mathcal{A})$;
  `FimSe`
`FimEnquanto`

Mostremos que, neste caso, a desigualdade também é válida para $i = k$, isto é, que $\pi^s_{k+1} \geq \pi^s_k$.

Seja π^s_k o rótulo do $k^{\underline{\text{ésimo}}}$ nó retirado de X', digamos $i \in X'$ tal que $h(i) = u$; consideremos também que π^s_{k+1} é o rótulo do $(k+1)^{\underline{\text{ésimo}}}$ nó retirado de X', digamos $j \in X'$ tal que $h(j) = v$. Admitamos que $\pi^s_{k+1} < \pi^s_k$.

No passo do algoritmo em que o nó i é retirado de X', o nó j ou estava em X' ou foi rotulado a partir de i.[3] Suponhamos que j estava em X' quando i foi retirado. Então i foi escolhido de modo a que $\pi^s_i \leq \pi^s_x$, para todo o $x \in X'$, donde $\pi^s_k \leq \pi^s_{k+1}$, o que contraria a hipótese. Concluímos então que j foi rotulado a partir de i, pelo que $\pi^s_j = \pi^s_i + c_{uv}$; isto é, $\pi^s_{k+1} = \pi^s_k + c_{uv}$. Como por hipótese os custos dos arcos são não negativos, então $\pi^s_{k+1} \geq \pi^s_k$. □

[3]Identificamos um passo do algoritmo com o ciclo "`Enquanto`".

Teorema 5.1 *Os trajectos $p_1, \ldots, p_K$, enumerados pela generalização do algoritmo de Dijkstra são determinados por ordem não decrescente de custos.*

Demonstração: O custo, $c(p_k)$, do trajecto p_k, é o rótulo de um nó da árvore que corresponde ao nó terminal da rede, para $1 \leq k \leq K$. Assim, $\{c(p_1), \ldots, c(p_K)\}$ é uma subsequência da sequência dos rótulos dos nós retirados de X', $\{\pi_1^s, \ldots, \pi_\ell^s\}$, mas contendo apenas os rótulos de nós da árvore correspondendo a t. Pelo lema 5.4, aquela sucessão é não decrescente, pelo que $c(p_1) \leq c(p_2) \leq \cdots \leq c(p_K)$, como pretendíamos. □

A utilização do algoritmo GD exige que a rede $(\mathcal{N}, \mathcal{A})$ não contenha **ciclos nulos**, isto é, ciclos de custo nulo. De facto, se tal não acontecer não é possível garantir que o algoritmo termine num número finito de passos, como é demonstrado no lema 5.5.

Lema 5.5 *Se $c(\mathcal{C}) = \sum_{\mathcal{C}} c_{ij} > 0$, para todo o ciclo $\mathcal{C}$ de $(\mathcal{N}, \mathcal{A})$, então a generalização do algoritmo de Dijkstra é finita.*

Demonstração: Suponhamos que o algoritmo GD não é finito e mostremos que então existe pelo menos um ciclo $\mathcal{C}$ em $(\mathcal{N}, \mathcal{A})$ tal que $c(\mathcal{C}) = 0$.

Se o algoritmo não é finito, ou seja, se o algoritmo não termina num número finito de passos, então o nó t não é rotulado K vezes e o conjunto X' nunca fica vazio. Como o número de nós e o número de caminhos numa rede é finito, existe pelo menos um ciclo $\mathcal{C}$ em $(\mathcal{N}, \mathcal{A})$, que é acrescentado à árvore em construção um número não finito de vezes. Este ciclo não contém o nó t dado que, se tal acontecesse, t seria rotulado K vezes e então o algoritmo terminaria.

Seja $i \in \mathcal{N}$ um dos nós de $\mathcal{C}$ e, consequentemente, um nó rotulado e escolhido em X' um número não finito de vezes. Por hipótese existe um caminho, p, de i para t, dado que $\mathcal{P}_{it} \neq \emptyset$, e portanto $p, \mathcal{C} \diamond p, \ldots, \mathcal{C}^k \diamond p$ são também trajectos de i para t em $(\mathcal{N}, \mathcal{A})$, com $k \geq 0$. Consideremos ainda, sem perda de generalidade, que o trajecto p e o ciclo $\mathcal{C}$ se separam imediatamente a seguir a i, isto é, que o segundo nó de p, digamos u, não coincide com o nó seguinte a i em $\mathcal{C}$, j.

Sejam $\{i_1, i_2, \ldots\}$ as ordens dos nós da árvore tais que $h(i_k) = i$, para $k \in \mathbb{N}$, relativas a um trajecto na árvore contendo esse ciclo. Então $\{\pi_{i_1}^s, \pi_{i_2}^s, \ldots\}$ é uma sucessão tal que $\pi_{i_k}^s = \pi_{i_1}^s + kc(\mathcal{C})$, para $k > 1$. Analogamente, o nó j é rotulado um número não finito de vezes, dado que também está em $\mathcal{C}$. Sejam $\{j_1, j_2, \ldots\}$ as ordens dos nós da árvore tais que $h(j_k) = j$, para $k \in \mathbb{N}$, relativas ao mesmo trajecto na árvore.

Admitindo que $c(\mathcal{C}) > 0$, então $\lim_{k \to +\infty} \pi_{i_k}^s = +\infty$, e como $\pi_{j_k}^s = \pi_{i_k}^s + c_{ij}$, com $k \in \mathbb{N}$, também $\lim_{k \to +\infty} \pi_{j_k}^s = +\infty$.

Suponhamos ainda que o nó u de p é rotulado e seja v o nó da árvore para o qual $h(v) = u$.

Dado que o ciclo $\mathcal{C}$ é acrescentado à árvore um número não finito de vezes, podemos concluir que p não é acrescentado. Como a partir de i são rotulados os nós j e u, e o escolhido é j, então $\pi^s_{j_k} \leq \pi^s_v = \pi^s_{i_1} + c_{iu}$, para todo o $k \in \mathbb{N}$, e $\lim_{k\to+\infty} \pi^s_{j_k} = +\infty$.

Sendo os custos dos arcos da rede finitos, π^s_v tem de ser finito, pelo que $c(\mathcal{C}) \leq 0$. Dado que admitimos que $(\mathcal{N}, \mathcal{A})$ não contém ciclos negativos concluímos, como pretendíamos, que $c(\mathcal{C}) = 0$. $\square$

Para exemplificar o resultado enunciado no lema 5.5 consideremos a rede da figura 9, onde $\mathcal{C} = \langle 1, 2, 3, 1\rangle$ é um ciclo nulo e onde $s = 1$ é o nó inicial e $t = 5$ é o terminal. Vamos em seguida "simular" o algoritmo apresentado quando aplicado à rede descrita naquela figura.

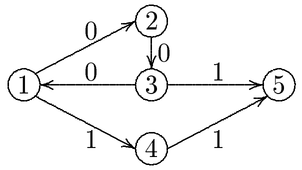

Figura 9

Inicialmente $X' = \{1\}$ e $h(1) = s = 1$. A partir de $1 \in X'$ são rotulados os nós 2 e 3, respectivamente com $\pi^s_2 = 0$ e $\pi^s_3 = 1$; 2 e 3 são inseridos em X'. Então, no final do primeiro passo, $X' = \{2, 3\}$, $h(2) = 2$ e $h(3) = 4$. Neste momento a árvore construída é a apresentada na figura 10. (É de notar que, como referimos atrás, nesta figura o índice do nó $i \in X$ é $h(i)$, para facilitar a identificação dos trajectos considerados.)

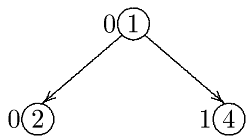

Figura 10

De seguida é retirado de X' o nó com menor rótulo, ou seja $i = 2$, e a partir deste nó é rotulado o nó 4, com $\pi^s_4 = 0$. No final deste passo, $X' = \{3, 4\}$ e $h(4) = 3$. Neste momento a árvore é a apresentada na figura 11.

No próximo passo do algoritmo 4 é retirado de X', e a partir de $h(4) = 3$ são rotulados os nós 5 e 6, com $\pi^s_5 = 0$ e $\pi^s_6 = 1$. Após este passo o conjunto X' é

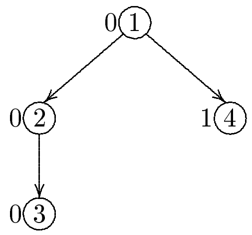

Figura 11

constituído por $X' = \{3, 5, 6\}$ com $h(5) = 1$ e $h(6) = 5$. A árvore neste momento é a apresentada na figura 12.

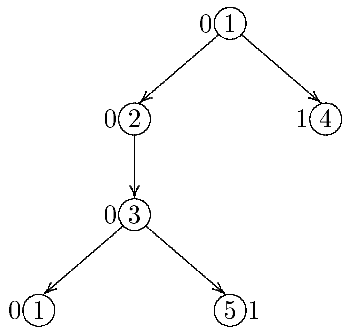

Figura 12

Podemos agora verificar facilmente que, continuando o algoritmo, seria retirado o nó $5 \in X'$, sendo repetidos sucessivamente o segundo e terceiro passos, nunca sendo encontrado qualquer trajecto de 1 para 5, isto é, nunca escolhendo em X' um nó associado ao terminal, $t = 5$. Concluímos pois que o algoritmo não terminaria se aplicado à rede da figura 9.

Note-se que a condição enunciada no lema 5.5 é suficiente mas não é necessária. De facto, o algoritmo pode ser finito e existir um ciclo nulo na rede. Para tal basta que nenhum nó de $\mathcal{C}$ seja escolhido em X' antes da determinação de p_K.

Tal pode ser verificado se considerarmos, por exemplo, a rede apresentada na figura 13, onde $s = 1$ e $t = 6$, notando que o ciclo $\mathcal{C} = \langle 2, 3, 5, 2 \rangle$ desta rede tem custo $c(\mathcal{C}) = 0$.

Se nesta rede pretendermos encontrar os $K = 2$ trajectos mais curtos de 1 para 6, o ciclo `Enquanto` do algoritmo é executado um número finito de vezes (cinco) e a árvore obtida será a apresentada na figura 14.

À semelhança do que acontecia no algoritmo de pesquisa exaustiva, para cada nó i da árvore é necessário guardar os valores de π_i^s, ξ_i^s e $h(i)$ e utilizar ainda um conjunto X' constituído por todos os nós da árvore.

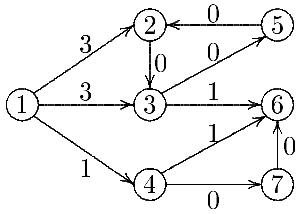

Figura 13

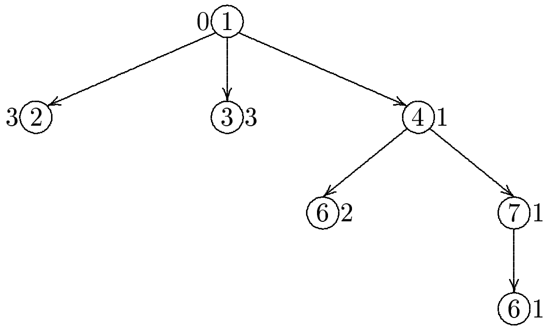

Figura 14

O ciclo `Enquanto` deve ser executado K vezes[4], mas, na pior das hipóteses, é necessário construir a árvore de todos os trajectos para t com raiz em s, tal como acontece no algoritmo de pesquisa exaustiva.

Além das rotulações realizadas, tantas quantos os nós da árvore construída, é ainda necessário escolher o nó em X' com menor rótulo, de cada vez que é executado um ciclo `Enquanto`. O conjunto X' pode ser mantido de forma ordenada ou desordenada. No segundo caso aquela escolha pode ser morosa se X' contiver muitos elementos, uma vez que será necessário efectuar uma pesquisa exaustiva do elemento com menor rótulo em X'.

Assim, a ordem de complexidade deste algoritmo é, no pior dos casos, a ordem de complexidade do algoritmo de pesquisa exaustiva. Esta ordem de complexidade depende, como foi referido, do número de nós da árvore dos trajectos construída e é, no pior dos casos, de $\mathcal{O}(n!)$.

A vantagem deste algoritmo em relação ao da pesquisa exaustiva é que pode suceder que não seja necessário realizar a pesquisa exaustiva, sendo construída apenas parte da árvore de todos os trajectos de s para t.

[4]O ciclo `Enquanto` só é executado menos de K vezes se o conjunto X' se tornar vazio, o que significa que o número de trajectos de s para t em $(\mathcal{N}, \mathcal{A})$ é inferior a K.

5.2.3 Generalização do Algoritmo de Yen

Neste parágrafo iremos descrever um algoritmo para a determinação ordenada dos K trajectos mais curtos de s para t numa rede, que se baseia ainda na construção da árvore dos trajectos de s para t, ou de parte dela.

Este algoritmo generaliza o algoritmo de Yen, [26, 27], para a enumeração dos K caminhos mais curtos, que será descrito na altura conveniente. No que se segue este algoritmo poderá ser designado abreviadamente por algoritmo GY.

De um modo genérico, podemos dizer que a ideia base desta generalização consiste em determinar sucessivos trajectos de s para t começando pelo mais curto e, a partir de cada um dos determinados, calcular outros. Cada um destes novos trajectos poderá ser um dos k mais curtos, pelo que dizemos que são candidatos ao $k^{\underline{\text{ésimo}}}$ trajecto mais curto, com $k \in \{2, \ldots, K\}$. O procedimento é repetido e com base no candidato de menor custo são calculados novos trajectos, candidatos ao trajecto mais curto seguinte.

Este algoritmo utiliza um conjunto, que designaremos por P, que contém os trajectos candidatos ao $k^{\underline{\text{ésimo}}}$ mais curto, com $1 \leq k \leq K$, ainda por analisar. De modo a obter a ordenação dos trajectos, em cada passo do algoritmo será retirado de P aquele que tiver custo mínimo. O trajecto retirado de P no $k^{\underline{\text{ésimo}}}$ passo é p_k, com $k \in \{1, \ldots, K\}$.

P designa pois um conjunto constituído pelos trajectos que vão sendo calculados e que não foram ainda analisados, enquanto que k indica o número de trajectos mais curtos determinados até um dado momento, ou seja, o número de trajectos analisados.

Consideremos que p_k é o $k^{\underline{\text{ésimo}}}$ trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$, com $k \in \{1, \ldots, K\}$, e que

$$p_k = \langle v_1^k, a_1^k, v_2^k, \ldots, a_{\ell_k - 1}^k, v_{\ell_k}^k \rangle.$$

No $k^{\underline{\text{ésimo}}}$ passo do algoritmo, p_k é escolhido como o elemento de P de menor custo, e a partir de p_k são calculados vários novos trajectos que se desviam de p_k em algum dos seus nós.

Seja $v_i^k \neq t$ um nó de p_k coincidente com o seu nó desvio ou com um dos nós que lhe seguem, isto é, $\alpha_k \leq i \leq \ell_k - 1$. É de notar que para $i = \ell_k$, v_i^k é o nó terminal e uma vez que assumimos que não existem arcos iniciados em t, basta analisar os nós coincidentes ou seguintes ao nó desvio de p_k mas anteriores ao terminal.

Consideremos que existem trajectos de v_i^k para t, cujo primeiro arco é diferente do arco a_i^k de p_k e seja q o mais curto desses trajectos. Então $\text{sub}_k(s, v_i^k) \diamond q$ é um trajecto de s para t, que coincide com p_k de s até v_i^k, mas que se separa de p_k nesse mesmo nó. Se obrigarmos ainda a que o primeiro arco de q não pertença à árvore dos trajectos com raiz em s construída até ao momento, obtemos um trajecto que denotaremos por q_i^k. Assim, o trajecto $p_i^k = \mathrm{sub}_k(s, v_i^k) \diamond q_i^k$, de s para t, é diferente de todos os anteriormente determinados, e é por isso um candidato a $r^{\underline{\text{ésimo}}}$ trajecto mais curto, com $r > k$.

Deste modo, a determinação de q_i^k, e consequentemente de p_i^k, depende do cálculo do trajecto mais curto de v_i^k para t, cujo primeiro arco não pertence à árvore dos trajectos construída até ao momento.

A ideia que surge de imediato para a determinação de q_i^k consiste em calcular simplesmente o trajecto mais curto de v_i^k para t, numa rede obtida de $(\mathcal{N}, \mathcal{A})$ removendo os arcos da árvore formada por $p_1, \ldots, p_k$, com início no nó dessa árvore associado ao nó v_i^k de p_k. No entanto este processo não é válido, uma vez que deste modo q_i^k poderia não conter qualquer desses arcos, o que poderia comprometer a possibilidade de determinação de alguns trajectos.

Para clarificar, consideremos por exemplo a rede da figura 15, onde os nós inicial e terminal são, respectivamente, $s = 1$ e $t = 3$.

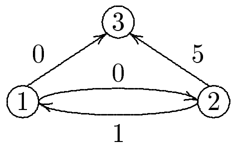

Figura 15

Nesta rede, o trajecto mais curto de 1 para 3 é $p_1 = \langle 1, 3 \rangle$. Além disso, o trajecto mais curto de 1 para 3 cujo primeiro arco não é $(1, 3)$ é o trajecto $p = \langle 1, 2, 1, 3 \rangle$, com custo $c(p) = 1$. No entanto, removendo o arco $(1, 3)$ de $(\mathcal{N}, \mathcal{A})$ e determinando então o trajecto mais curto de 1 para 3 obtemos $q = \langle 1, 2, 3 \rangle$, que tem custo $c(q) = 5$, superior ao custo de p.

Iremos em seguida descrever uma forma de calcular o trajecto mais curto entre dois nós, cujo primeiro arco não pertence a um dado conjunto.

Dado $k \in \{1, \ldots, K\}$, denotemos por $\mathcal{P}_k = \{p_1, \ldots, p_k\}$ o conjunto dos k trajectos mais curtos de s para t. O conjunto $P \cup \mathcal{P}_k$ contém todos os trajectos determinados até ao passo k; nomeadamente, os k trajectos mais curtos, elementos de $\mathcal{P}_k$, e os candidatos a $r^{\underline{\text{ésimo}}}$ mais curto, com $r > k$, determinados até ao momento, elementos de P. Em suma, o conjunto $P \cup \mathcal{P}_k$ contém todos os trajectos que constituem a árvore dos trajectos de s para t, no passo k do algoritmo.

Consideremos que X^k e T^k são, respectivamente, o conjunto dos nós e o conjunto dos arcos na árvore dos trajectos de s para t, no final da execução do passo k do algoritmo; isto é,

$$X^k = \{u \in X : h(u) \text{ é nó de } p \in P \cup \mathcal{P}_k\}$$

e

$$T^k = \{(u,v) \in X^k \times X^k : (h(u), h(v)) \text{ é arco de } p \in P \cup \mathcal{P}_k\}.$$

É de notar que $X^k \subseteq X$ é um subconjunto próprio de $\mathbb{N}$ e que $h(X^k)$ é um subconjunto de $\mathcal{N}$. É de notar ainda que podemos considerar uma aplicação associada a h, a que chamaremos g, que relaciona os arcos da árvore com os arcos na rede, da seguinte forma

$$g : X \times X \longrightarrow \mathcal{N} \times \mathcal{N} : (i,j) \longrightarrow g(i,j) = (h(i), h(j)).$$

Ao considerar um trajecto da árvore com raiz em s referimo-nos a um trajecto p em $(\mathcal{N}, \mathcal{A})$ ou ao trajecto constituído pelos nós e arcos associados aos nós e arcos de p na árvore em questão, dependendo do contexto. Estes dois trajectos serão pois identificados por vezes um com o outro.

Dado um nó $u \in X^k$, denotemos por $\mathcal{A}^k(u) = \{(u,v) \in T^k : v \in X^k\}$ o conjunto dos arcos na árvore obtida no passo k do algoritmo, com início naquele nó, e por $\mathcal{D}^k(u) = \{v \in X^k : (u,v) \in \mathcal{A}^k(u)\}$ o conjunto dos nós da árvore sucessores dos arcos de $\mathcal{A}^k(u)$.

Voltemos a considerar que v_i^k é um nó de $p_k = \langle v_1^k, \ldots, v_{\ell_k}^k \rangle$ e que $u \in X^k$ é o nó da árvore correspondente a v_i^k, tal que $h(u) = v_i^k$. Denotemos por $P^k(i)$ o conjunto constituído pelos trajectos em $(\mathcal{N}, \mathcal{A})$ de v_i^k para t, cujo primeiro arco não pertence ainda à árvore dos trajectos no passo k do algoritmo. Os elementos de $P^k(i)$ são trajectos da forma $\langle v_i^k, j \rangle \diamond q_j$, onde j é um nó da rede $(\mathcal{N}, \mathcal{A})$, q_j é um trajecto em $(\mathcal{N}, \mathcal{A})$ desde esse nó até ao nó terminal t, e onde o arco (v_i^k, j) é um arco da rede não pertencente à árvore dos trajectos no passo k do algoritmo; isto é, $(v_i^k, j) \in \mathcal{A} - g(\mathcal{A}^k(u))$, ou ainda $j \in \mathcal{D}(v_i^k) - h(\mathcal{D}^k(u))$.

Para determinarmos q_i^k deveremos pois determinar o trajecto mais curto em $P^k(i)$, podendo para tal utilizar o lema 5.6.

Lema 5.6 *O trajecto de menor custo de entre os elementos do conjunto $P^k(i)$ é da forma $q_i^k = \langle v_i^k, j \rangle \diamond q_j$, onde $j \in \mathcal{D}(v_i^k) - h(\mathcal{D}^k(u))$ e onde q_j é o trajecto mais curto de j para t em $(\mathcal{N}, \mathcal{A})$.*

Demonstração: Pela definição de $P^k(i)$, todos os seus elementos são da forma $q = \langle v_i^k, j \rangle \diamond q_j$, onde $j \in \mathcal{D}(v_i^k) - h(\mathcal{D}^k(u))$ e $q_j \in \mathcal{P}_{jt}$.

Mostremos que se q é o trajecto de menor custo em $P^k(i)$, q_j é o trajecto mais curto de j para t em $(\mathcal{N}, \mathcal{A})$.

Suponhamos que q_j não é o trajecto mais curto de j para t, isto é, existe um outro trajecto p_j em $\mathcal{P}_{jt}$ tal que $c(p_j) < c(q_j)$. Assim, $p = \langle v_i^k, j\rangle \diamond p_j$ é um trajecto de $P^k(i)$ e, por hipótese,

$$c(p) = c_{v_i^k j} + c(p_j) < c_{v_i^k j} + c(q_j) = c(q),$$

pelo que então q não seria o mais curto em $P^k(i)$. $\square$

Se no início do algoritmo for construída a árvore dos trajectos mais curtos de todos os nós para t, conhecemos também o custo de todos esses trajectos. Deste modo, para encontrar o trajecto q_i^k basta procurar o arco (v_i^k, j), não pertencente à árvore dos trajectos no passo k do algoritmo, que minimiza $c_{v_i^k j} + c(q_j)$, onde q_j é o trajecto mais curto de j para t e que pertence à árvore construída no início do algoritmo.

No que se segue denotaremos por $\mathcal{T}_t$ a árvore dos trajectos mais curtos de todos os nós para t. Dados dois nós i e j da rede, iremos ainda representar por $\mathcal{T}_t(i,j)$ o trajecto de i para j na árvore $\mathcal{T}_t$ e por $\mathcal{T}_t(i)$ o trajecto em $\mathcal{T}_t$ de i para t. Analogamente, $\mathcal{T}_s$ representará a árvore dos trajectos mais curtos de s para os restantes nós de $(\mathcal{N}, \mathcal{A})$, $\mathcal{T}_s(i,j)$ e $\mathcal{T}_s(i)$ representarão, respectivamente os trajectos de i para j e de s para i em $\mathcal{T}_s$. É de notar que nem sempre existem os trajectos $\mathcal{T}_s(i,j)$ e $\mathcal{T}_t(i,j)$ nas árvores $\mathcal{T}_s$ e $\mathcal{T}_t$, respectivamente.

Resumindo, sabemos então que o trajecto obtido a partir do nó v_i^k de p_k é dado por $p_i^k = \mathrm{sub}_k(s, v_i^k) \diamond q_i^k$, onde q_i^k é o trajecto mais curto do conjunto $P^k(i)$ e é da forma $q_i^k = \langle v_i^k, j\rangle \diamond \mathcal{T}_t(j)$, com $j \in \mathcal{D}(v_i^k) - h(\mathcal{D}^k(u))$.

Este é um novo trajecto, isto é, diferente de todos os outros pertencentes à árvore com raiz em s construída até ao passo k do algoritmo, como é demonstrado no teorema 5.2.

Teorema 5.2 *Seja q_i^k o trajecto mais curto no conjunto $P^k(i)$. Então o trajecto $p_i^k = sub_k(s, v_i^k) \diamond q_i^k$ não pertence a $\mathcal{P}_k \cup P$, isto é, não foi determinado em nenhum passo anterior ao $k^{\underline{\text{ésimo}}}$.*

Demonstração: Seja $p_i^k = \mathrm{sub}_k(s, v_i^k) \diamond q_i^k$, onde q_i^k é o trajecto mais curto no conjunto $P^k(i)$. Então p_i^k coincide com p_k desde o nó inicial, s, até ao nó v_i^k.

Por definição, $q_i^k = \langle v_i^k, j\rangle \diamond \mathcal{T}_t(j) \in P^k(i)$, com $j \in \mathcal{D}(v_i^k) - h(\mathcal{D}^k(u))$, onde u é o nó da árvore associado ao nó v_i^k de p_k.

Mas então o trajecto $\mathrm{sub}_k(s, v_i^k) \diamond \langle v_i^k, j\rangle$ não pertence à árvore formada pelos trajectos em P e por $p_1, \ldots, p_k$, donde $p_i^k \notin \mathcal{P}_k \cup P$, como pretendíamos mostrar. $\square$

É ainda de realçar que se o conjunto $P^k(i)$ for vazio não é possível definir novos trajectos nas condições que foram referidas, pelo que nesse caso não é acrescentado nenhum candidato ao conjunto P.

De acordo com o raciocínio apresentado é possível obter uma generalização do algoritmo de Yen, que enumera os K trajectos mais curtos entre dois nós de uma rede. Este raciocínio é descrito no algoritmo 5.3.

Algoritmo 5.3: *Generalização do algoritmo de Yen*

$\mathcal{T}_t \longleftarrow$ árvore dos trajectos mais curtos de todos os nós para t em $(\mathcal{N}, \mathcal{A})$;
$p \longleftarrow \mathcal{T}_t(s)$ (trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$);
$P \longleftarrow \{p\}$;
$k \longleftarrow 1$;
`Enquanto` $(k \leq K)$ e $(P \neq \emptyset)$ `Fazer`
    $p_k \longleftarrow$ trajecto de P tal que $c(p_k) \leq c(p)$, qualquer que seja $p \in P$;
    $P \longleftarrow P - \{p_k\}$;
    $\alpha_k \longleftarrow$ ordem do nó desvio de p_k;
    `Para` (todo o $i \in \{\alpha_k, \ldots, l_k - 1\}$) `Fazer`
        $X \longleftarrow \{\langle v_i^k, j\rangle \diamond \mathcal{T}_t(j) : (v_i^k, j)$ não pertence à árvore construída$\}$;
        `Se` $(X \neq \emptyset)$ `Então`
            $q_i^k \longleftarrow$ trajecto mais curto em X;
            $p_i^k \longleftarrow \mathrm{sub}_k(s, v_i^k) \diamond q_i^k$;
            $P \longleftarrow P \cup \{p_i^k\}$;
        `FimSe`
    `FimPara`
    $k \longleftarrow k + 1$;
`FimEnquanto`

É demonstrado, no teorema 5.3, que segundo este algoritmo os trajectos $p_1, \ldots, p_K$ são determinados ordenadamente.

Teorema 5.3 *Na generalização do algoritmo de Yen os trajectos $p_1, \ldots, p_K$ são enumerados por ordem não decrescente de custos.*

Demonstração: Sejam $p_1, \ldots, p_K$ os trajectos de s para t em $(\mathcal{N}, \mathcal{A})$, determinados pela generalização do algoritmo de Yen. Mostremos, por indução sobre k, que $c(p_k) \leq c(p_{k+1})$, para $1 \leq k < K$.

Para $k = 1$, p_1 é o trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$, e portanto, por definição, $c(p_1) \leq c(p)$ para todo o $p \in \mathcal{P}$, logo $c(p_1) \leq c(p_2)$.

Suponhamos que p_k é o trajecto retirado de P no $k^{\underline{\text{ésimo}}}$ passo do algoritmo e mostremos que $c(p_k) \leq c(p_{k+1})$, com $k > 1$.

Suponhamos então que $c(p_{k+1}) < c(p_k)$.

Se no $k^{\underline{\text{ésimo}}}$ passo do algoritmo p_{k+1} pertencia a P, então, como por hipótese $c(p_{k+1}) < c(p_k)$, o trajecto escolhido deveria ter sido p_{k+1} e não p_k.

Se o trajecto p_{k+1} não pertencia a P, então foi calculado a partir de p_k, ou seja, $p_{k+1} = \text{sub}_k(s, v_i^k) \diamond q_i^k$, com $\alpha_k \leq i \leq l_k - 1$, onde q_i^k é o trajecto mais curto de v_i^k, nó de p_k, para t, cujo primeiro arco não pertence à árvore contendo os trajectos de $\mathcal{P}_k \cup P$.

Como $k > 1$, o trajecto p_k foi obtido a partir de um outro, digamos p_r, com $1 \leq r < k$. Ou seja,

$$p_k = \text{sub}_k(s, v_i^k) \diamond \text{sub}_k(v_i^k, t) = \text{sub}_r(s, v_{\alpha_k}^k) \diamond \text{sub}_k(v_{\alpha_k}^k, t).$$

Se $\alpha_k < i$, então

$$p_{k+1} = \text{sub}_k(s, v_i^k) \diamond q_i^k = \text{sub}_r(s, v_{\alpha_k}^k) \diamond \text{sub}_k(v_{\alpha_k}^k, v_i^k) \diamond q_i^k;$$

se $\alpha_k = i$, então o nó $v_{\alpha_k}^k$ coincide com o nó v_i^k, e

$$p_{k+1} = \text{sub}_k(s, v_i^k) \diamond q_i^k = \text{sub}_k(s, v_{\alpha_k}^k) \diamond q_i^k.$$

Em qualquer dos casos, quer p_k quer p_{k+1} contêm subtrajectos de $v_{\alpha_k}^k$ até t. Designemos estes dois subtrajectos por q_k e q_{k+1}, respectivamente.

Como, por hipótese, $c(p_{k+1}) < c(p_k)$ e p_k coincide com p_{k+1} desde s até $v_{\alpha_k}^k$, então também $c(q_{k+1}) < c(q_k)$.

Por outro lado, pelo algoritmo apresentado, q_k é o trajecto mais curto em $P^r(\alpha_k)$. Tentemos pois mostrar que, além disso, $q_{k+1} \in P^r(\alpha_k)$.

Se $\alpha_k < i$, o primeiro arco de q_{k+1} coincide com o de q_k e portanto $q_{k+1} \in P^r(\alpha_k)$.

Se $\alpha_k = i$, então q_{k+1} coincide com q_i^k, onde q_i^k é um elemento de $P^k(i)$, o que significa que o primeiro arco de q_i^k não pertence à árvore construída no passo k do algoritmo; logo também não pertence à árvore que foi construída no $r^{\underline{\text{ésimo}}}$ passo, uma vez que $r < k$.

Em qualquer dos casos concluímos, como pretendíamos, que $q_{k+1} \in P^r(\alpha_k)$, pelo que q_k não seria o trajecto mais curto em $P^r(\alpha_k)$, aquando do cálculo de p_k. $\square$

Uma vez que os trajectos são determinados de forma ordenada, o algoritmo irá terminar assim que forem encontrados os K primeiros trajectos, ou seja, os K mais curtos. Se o número de trajectos de s para t em $(\mathcal{N}, \mathcal{A})$ for inferior a K o algoritmo termina assim que tiverem sido calculados todos os trajectos da rede.

O resultado que se segue é uma consequência imediata do teorema 5.3, pelo que é omitida a sua demonstração.

Corolario 5.3.1 *Sejam $p_1, \ldots, p_K$ os trajectos determinados pela generalização do algoritmo de Yen. Então, qualquer que seja o trajecto p de $\mathcal{P} - \{p_1, \ldots, p_K\}$, $c(p_K) \leq c(p)$.*

É de notar que o conjunto P contém trajectos candidatos a $k^{\underline{\text{ésimo}}}$ trajecto mais curto para valores de k em $\{1, \ldots, K\}$. Ou seja, uma vez determinados k trajectos, é suficiente que se guardem apenas os $K - k$ melhores trajectos candidatos obtidos até à execução daquele passo.

Para ilustrar a execução deste algoritmo, vamos apresentar alguns dos seus passos quando aplicado à determinação dos K trajectos mais curtos de $s = 1$ para $t = 6$ na rede $(\mathcal{N}, \mathcal{A})$, representada na figura 13.

Antes de mais é calculada a árvore $\mathcal{T}_t$ dos trajectos mais curtos de todos os nós para t, representada na figura 16.

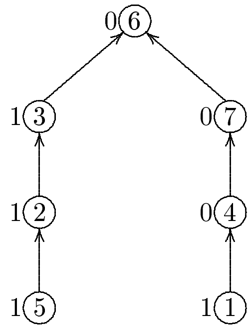

Figura 16

Após ter sido determinada esta árvore, é iniciado o cálculo dos trajectos mais curtos de 1 para 6. O primeiro trajecto calculado é $p_1 = \langle 1, 4, 7, 6 \rangle$, que passa a ser elemento de P. No primeiro passo do algoritmo, p_1 é retirado de P e a partir de p_1 são calculados alguns trajectos. No final deste passo a árvore de trajectos de 1 para 6 é a representada na figura 17. É de notar que esta árvore é constituída por p_1 e por candidatos a p_2. É ainda de notar, que para determinar o trajecto $\langle 1, 2, 3, 6 \rangle$ desta árvore é necessário calcular o mínimo entre $c(\langle 1, 2 \rangle) + c(\mathcal{T}_t(2)) = 4$ e $c(\langle 1, 3 \rangle) + c(\mathcal{T}_t(3)) = 4$.

No passo seguinte da execução da generalização do algoritmo de Yen, isto é, no final do segundo passo a árvore mantém-se inalterada uma vez que a partir do segundo trajecto mais curto, $p_2 = \langle 1, 4, 6 \rangle$, não são calculados novos trajectos. De facto, sendo 4 o nó desvio de p_2, será necessário determinar o trajecto mais curto de 4 para 6, cujo primeiro arco não seja $(4, 6)$ nem $(4, 7)$. Uma vez que não existe

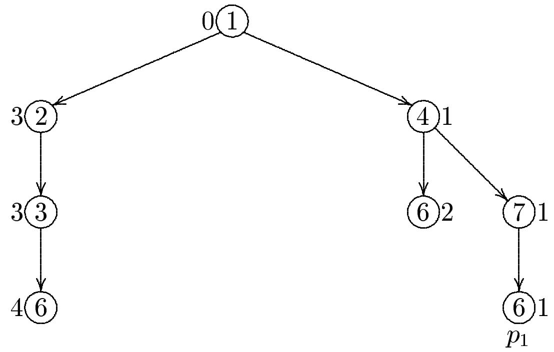

Figura 17

nenhum trajecto nestas condições, somente após o terceiro passo obtemos a árvore representada na figura 18.

Apesar de terminarmos aqui esta simulação, o algoritmo poderia continuar até que tivessem sido executados K passos, para qualquer valor de K. Isto acontece uma vez que nesta rede pode ser definido um número não finito de trajectos entre 1 e 6, dada a existência do ciclo $\langle 2, 3, 5, 2 \rangle$.

Como foi referido, e como é possível constatar nas figuras 17 e 18, neste algoritmo, dado um novo trajecto p_k, é calculado um trajecto para o seu nó desvio e seguintes. Para cada um daqueles nós, é acrescentado um novo ramo ou parte dele à árvore dos trajectos. Podemos por isso dizer que a construção da árvore se realiza em "profundidade".

Neste algoritmo, caso existam, são determinados K trajectos mais curtos de s para t, sendo o ciclo `Enquanto` nesse caso executado K vezes.

Cada execução do ciclo implica uma pesquisa no conjunto P dos candidatos a $k^{\underline{\text{ésimo}}}$ mais curto, com $1 \leq k \leq K$, e a determinação de, quando muito, $n - 1$ trajectos que são desvios.

O conjunto P, dos candidatos a $k^{\underline{\text{ésimo}}}$ trajecto mais curto, com $k \in \{1, \ldots, K\}$, contém, no máximo, K elementos. Por utilização do método *adressed calculation sort* de Dijkstra, [6], a pesquisa do trajecto com menor custo em P pode ser realizada utilizando um número constante de operações, não influenciando portanto a ordem de complexidade do algoritmo GY.

O pior dos casos para a generalização do algoritmo de Yen ocorre para uma rede do tipo da representada na figura 19, em que todo o trajecto mais curto passa por todos os nós e onde, além disso existe um arco com origem em t e que termina em s, que designamos em geral por arco de retorno, com custo nulo.

Consideremos então que p_1 é o trajecto mais curto na rede da figura 19, e que

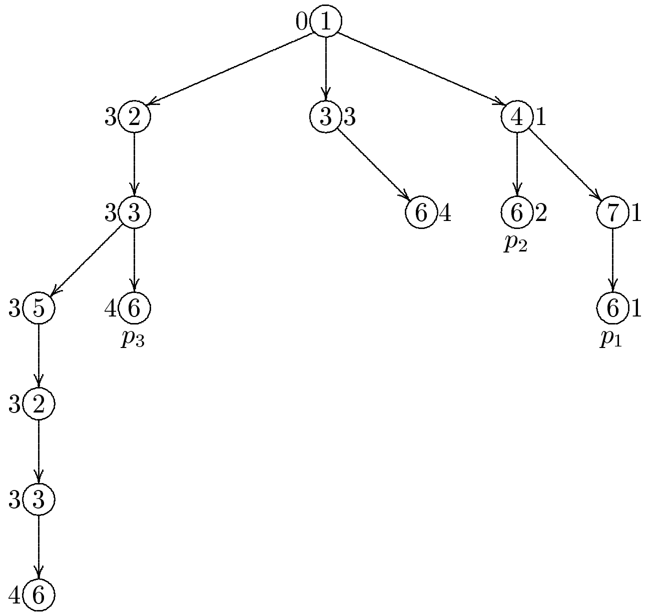

Figura 18

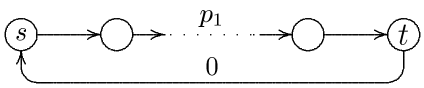

Figura 19

passa por todos os nós dessa rede. O $k^{\underline{\text{ésimo}}}$ trajecto mais curto é $p_k = p_{k-1} \diamond \langle t, s \rangle \diamond p_1$, para $k \geq 2$, que foi obtido a partir de p_{k-1}, utilizando o arco de retorno e voltando a realizar o trajecto p_1.

Assim, a determinação de um desvio consiste na pesquisa do trajecto com menor custo num conjunto $P^k(i)$, para um certo $v_i^k \in \mathcal{N}$, o que obriga à realização de uma pesquisa exaustiva no conjunto dos arcos com início em v_i^k. Esta pesquisa exige, no pior dos casos, que seja necessário analisar todos os arcos da rede, o que implica que sejam executadas m operações, para cada trajecto p_k.

O número total de operações realizadas pelo algoritmo é então Km, no pior dos casos, pelo que a generalização do algoritmo de Yen é de complexidade $\mathcal{O}(Km)$. A este valor é ainda de acrescentar o número de operações da construção da árvore $\mathcal{T}_t$, que é, no pior dos casos, de ordem $\mathcal{O}(n^2)$, se utilizarmos o algoritmo de Dijkstra, apresentado do parágrafo 3.2.

Em termos do espaço de memória ocupado, além do que é utilizado para representar a rede, é necessário ainda representar K trajectos, a árvore dos trajectos mais curtos de todos os nós para t e saber, para cada nó i de X^k, com $1 \leq k \leq K$, quais os arcos com início em i no conjunto T^k. O espaço de memória utilizado neste algoritmo é pois $Kn + n + Kn^2$, ou seja, da $\mathcal{O}(Kn^2)$.

5.2.4 Algoritmo de Eppstein

Neste parágrafo iremos descrever um algoritmo para a enumeração dos K trajectos mais curtos de s para t em $(\mathcal{N}, \mathcal{A})$, que foi recentemente proposto por Eppstein, [9]. Este algoritmo, tal como os apresentados anteriormente nesta subsecção, é baseado na construção da árvore dos trajectos entre s e t ou apenas de parte dela. A forma como é realizada esta construção apresenta algumas semelhanças com a generalização do algoritmo de Yen, descrita no parágrafo anterior.

A ideia base deste algoritmo continua pois a ser a determinação de sucessivos trajectos de s para t, a partir de outros previamente determinados.

É utilizado um conjunto, que designamos por P, constituído pelos candidatos a $k^{\underline{\text{ésimo}}}$ trajecto mais curto, com $k \in \{1, \ldots, K\}$.

Inicialmente é determinado o trajecto mais curto de s para t e $P = \{p_1\}$. De um modo genérico, no $k^{\underline{\text{ésimo}}}$ passo do algoritmo, para $k \in \{1, \ldots, K\}$, é retirado o trajecto com menor custo em P, trajecto esse que é p_k, o $k^{\underline{\text{ésimo}}}$ trajecto mais curto de s para t. Consideraremos no que se segue que p_k é da forma

$$p_k = \langle v_1^k, a_1^k, v_2^k, \ldots, a_{\ell_k - 1}^k, v_{\ell_k}^k \rangle.$$

A partir de p_k são determinados outros trajectos, que se desviam de p_k num certo nó, trajectos esses que passam a ser elementos de P, uma vez que são candidatos a $r^{\underline{\text{ésimo}}}$ trajecto mais curto, para algum $r > k$.

Este procedimento é análogo ao utilizado na generalização do algoritmo de Yen. A diferença entre estes dois algoritmos reside no facto de em cada um deles serem calculados diferentes trajectos a partir de cada p_k, como descreveremos em seguida.

Como referimos, o primeiro trajecto determinado é p_1. Este trajecto passa a ser elemento de P, de onde é retirado no primeiro passo do algoritmo, uma vez que é o seu único elemento nesse momento.

Comecemos então por considerar o trajecto mais curto, p_1. A partir de p_1 vamos calcular novos trajectos, entre os quais deverá estar p_2, que coincide com p_1 até um dos seus nós.

Seja v_i^1 um dos nós de p_1, tal que $1 \leq i \leq \ell_1 - 1$. Podemos determinar vários novos trajectos de s para t que coincidam com p_1 desde s até v_i^1, e que se separam dele nesse mesmo nó, sendo todos eles diferentes de p_1. Para isso basta calcular trajectos de v_i^1 para t, efectuando então a sua concatenação com $\text{sub}_1(s, v_i^1)$. De modo a não obter novamente p_1, os trajectos calculados de v_i^1 para t não devem coincidir com $\text{sub}_1(v_i^1, t)$.

Uma vez que pretendemos encontrar os trajectos com menor custo possível, devemos procurar trajectos entre os nós v_i^1 e t com o menor custo, à excepção de $\text{sub}_1(v_i^1, t)$. Os trajectos que se separam de $\text{sub}_1(v_i^1, t)$ no nó v_i^1 são então da forma $p = \langle v_i^1, j \rangle \diamond q_j$, onde (v_i^1, j) é um arco de $(\mathcal{N}, \mathcal{A})$ e q_j é um trajecto de j para t. Se $j \neq v_{i+1}^1$, isto é, se $(v_i^1, j) \not\equiv a_i^1$, então p não coincide com $\text{sub}_1(v_i^1, t)$, como pretendíamos. Podemos então procurar trajectos da forma $p = \langle v_i^1, j \rangle \diamond q_j$, com $(v_i^1, j) \in \mathcal{A} - \{(v_i^1, v_{i+1}^1)\}$, $q_j \in \mathcal{P}_{jt}$ e com o menor custo. Para isso iremos escolher os trajectos tais que q_j tem o menor custo possível. Como não é imposta nenhuma restrição a este último trajecto, q_j pode ser escolhido simplesmente como sendo o trajecto mais curto de j para t em $(\mathcal{N}, \mathcal{A})$, ou seja, utilizando a notação introduzida no parágrafo anterior, $q_j = \mathcal{T}_t(j)$.

Lembremos que dado um nó $i \in \mathcal{N}$, $\mathcal{D}(i)$ denota o conjunto dos nós da rede $(\mathcal{N}, \mathcal{A})$ nos quais termina um arco com início no nó i.

Resumindo, cada nó v_i^1 de p_1, com $1 < i < \ell_1$, dá origem aos trajectos em $(\mathcal{N}, \mathcal{A})$ da forma

$$\text{sub}_1(s, v_i^1) \diamond \langle v_i^1, j \rangle \diamond \mathcal{T}_t(j),$$

para todo o nó j do conjunto $\mathcal{D}(v_i^1) - \{v_{i+1}^1\}$. Para o caso particular de $i = 1$ obtemos, de modo análogo, os trajectos da forma $\langle s, j \rangle \diamond \mathcal{T}_t(j)$, com $j \in \mathcal{D}(s) - \{v_2^1\}$.

É de notar que na generalização do algoritmo de Yen procurávamos apenas, para cada nó v_i^1, o trajecto da forma indicada com menor custo.

O modo como é tratado o trajecto p_k, com $1 < k \leq K$, é análogo ao utilizado para $k = 1$ e será descrito em seguida.

De um modo geral iremos determinar vários trajectos a partir de cada trajecto $k^{\underline{\text{ésimo}}}$ mais curto, p_k, com $1 < k \leq K$.

Como $k > 1$, p_k foi obtido a partir de um certo trajecto p_r, com $1 \leq r < k$. Então p_k coincide com p_r desde s até um nó que é o seu nó desvio, $v_{\alpha_k}^r$, ou seja, $\text{sub}_k(s, v_{\alpha_k}^k) = \text{sub}_r(s, v_{\alpha_k}^r)$.

Devemos em seguida determinar trajectos que se desviem de p_k em algum dos seus nós. No entanto, quando p_k foi calculado, ou seja, quando p_r, que o originou, foi analisado, foram determinados os trajectos que se desviavam de p_r, em alguns dos seus nós anteriores a $v_{\alpha_k}^k$. Ao analisar p_k, para que não sejam calculados novamente alguns trajectos, basta determinar apenas aqueles que se separem de p_k na parte não comum com p_r, ou seja, em nós seguintes ao seu nó de desvio, $v_{\alpha_k}^k$, isto é, em cada v_i^k, com $\alpha_k < i < \ell_k$.[5]

[5]Novamente, pela razão indicada no parágrafo anterior, não é necessário analisar t, o último nó

Como anteriormente, para cada um desses nós v_i^k são determinados os trajectos da forma

$$\text{sub}_k(s, v_i^k) \diamond \langle v_i^k, j\rangle \diamond q_j,$$

onde j é um nó qualquer da rede tal que $j \in \mathcal{D}(v_i^k) - \{v_{i+1}^k\}$. Mais uma vez, de modo a determinar os trajectos com o menor custo possível escolhemos trajectos q_j tais que $q_j = \mathcal{T}_t(j)$.

É de notar que, como foi descrito, o primeiro nó a analisar a partir de p_1 deve ser o inicial, s, e a partir de p_k, para $k > 1$, deve ser $v_{\alpha_k+1}^k$. Assim, por conveniência, neste algoritmo iremos considerar que a ordem do nó desvio do trajecto p_1 é $\alpha_1 = 0$.

Fica assim concluída a descrição sumária do raciocínio seguido no algoritmo de Eppstein.

Para além do que foi referido, e com o intuito de simplificar a implementação computacional do algoritmo, Eppstein utiliza ainda uma mudança de variável, que será descrita no que se segue.

Tal como no algoritmo de Yen generalizado, no início deste algoritmo é determinada a árvore dos trajectos mais curtos de todos os nós da rede para t. Esta árvore é mantida durante a resolução do problema, o que permite conhecer de imediato os trajectos $\mathcal{T}_t(j)$, de j para t, com $j \in \mathcal{N} - \{t\}$, necessários. Além disso, a árvore $\mathcal{T}_t$ é necessária para a realização da mudança de variável.

Sejam i e j dois nós de $(\mathcal{N}, \mathcal{A})$ e (i, j) um arco da mesma rede. Denotemos, mais uma vez, por π_i^t o custo do trajecto mais curto de i para t na rede $(\mathcal{N}, \mathcal{A})$, isto é, $\pi_i^t = c(\mathcal{T}_t(i))$.

Associemos a cada arco (i, j) da rede o valor $\bar{c}_{ij} = \pi_j^t - \pi_i^t + c_{ij}$ que designaremos por **custo reduzido** de (i, j).

Por analogia com a noção de custo de um trajecto podemos considerar a de custo reduzido de um trajecto, definido por $\bar{c}(p) = \sum_p \bar{c}_{ij}$.

Como referimos anteriormente, a figura 16 representa graficamente a árvore dos trajectos mais curtos de todos os nós para t na rede que é representada na figura 13. Esta árvore é utilizada para calcular os custos reduzidos de cada arco. Para exemplificar a noção de custo reduzido que foi apresentada, a figura 20 mostra a rede da figura 13, onde o custo usual de cada arco foi substituído pelo respectivo custo reduzido.

Como podemos verificar pela observação daquelas figuras, o custo reduzido de qualquer arco da rede é não negativo. Se o arco em questão pertencer à árvore $\mathcal{T}_t$ podemos ainda observar que o respectivo custo reduzido é nulo. Esta propriedade é válida em qualquer rede, sendo demonstrada em seguida.

de p_k.

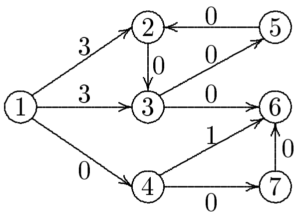

Figura 20

Lema 5.7 *Seja π_i^t a distância do trajecto mais curto de i para t numa dada rede $(\mathcal{N}, \mathcal{A})$ e seja $\bar{c}_{ij} = \pi_j^t - \pi_i^t + c_{ij}$. Então $\bar{c}_{ij} \geq 0$, qualquer que seja $(i,j) \in \mathcal{A}$, e $\bar{c}_{ij} = 0$, qualquer que seja (i,j) arco da árvore dos trajectos mais curtos de todos os nós para t, em $(\mathcal{N}, \mathcal{A})$.*

Demonstração: Seja $p_i = \langle i,j \rangle \diamond \mathcal{T}_t(j)$ um trajecto de i para t em $(\mathcal{N}, \mathcal{A})$.

Uma vez que $\mathcal{T}_t(i)$ é o trajecto de menor custo em $\mathcal{P}_{it}$, $c(\mathcal{T}_t(i)) \leq c(p_i)$. Por outro lado $c(\mathcal{T}_t(i)) = \pi_i^t$ e $c(p_i) = c_{ij} + \pi_j^t$, logo $c(p_i) - c(\mathcal{T}_t(i)) \geq 0$, ou seja, $\bar{c}_{ij} \geq 0$.

Suponhamos agora que (i,j) pertence à árvore dos trajectos mais curtos com raiz em t. Então p_i coincide com $\mathcal{T}_t(i)$, tendo-se $c(p_i) = c(\mathcal{T}_t(i))$, pelo que $\bar{c}_{ij} = 0$, como pretendíamos mostrar. □

O recíproco deste lema nem sempre é válido. De facto, como podemos verificar nas figuras 16 e 20, podem existir arcos que não pertençam à árvore $\mathcal{T}_t$, embora os seus custos reduzidos sejam nulos. Este é o caso, por exemplo, do arco $(3,5)$.

O custo reduzido $\bar{c}_{ij}$, de um arco (i,j), representa o custo acrescentado ao custo do trajecto mais curto de i para t, $\mathcal{T}_t(i)$, quando pretendemos seguir $p_i = \langle i,j \rangle \diamond \mathcal{T}_t(j)$, ou seja, quando nos desviamos de $\mathcal{T}_t(i)$ apenas no arco (i,j).

Consideremos por exemplo $p_1 = \langle 1,4,7,6 \rangle$, o trajecto mais curto de 1 para 6, na rede da figura 20. O trajecto $p = \langle 1,3,6 \rangle$, onde $\mathcal{T}_6(3) = \langle 3,6 \rangle$ é o trajecto com menor custo de 3 para 6, e embora tenha início em 1 e termine em 6, é diferente de p_1. Então, pelo que foi enunciado, o custo de p é dado por $c(p) = \bar{c}_{1,3} + c(p_1) = 4$. Este resultado é demonstrado, no lema 5.8.

Lema 5.8 *Seja (i,j) um arco de $(\mathcal{N}, \mathcal{A})$, e sejam $p_i = \langle i,j \rangle \diamond \mathcal{T}_t(j)$ e $\mathcal{T}_t(i)$ dois trajectos de i para t naquela rede. Então $c(p_i) = \bar{c}_{ij} + c(\mathcal{T}_t(i))$.*

Demonstração: Sob as hipóteses enunciadas podemos concluir que

$$c(p_i) - c(\mathcal{T}_t(i)) = c_{ij} + c(\mathcal{T}_t(j)) - c(\mathcal{T}_t(i)) = c_{ij} + \pi_j^t - \pi_i^t$$

e, então, pela definição de custo reduzido, $c(p_i) - c(\mathcal{T}_t(i)) = \bar{c}_{ij}$, como pretendíamos demonstrar. □

Como é demonstrado no lema 5.9, conhecendo o custo do trajecto mais curto de s para t, em $(\mathcal{N}, \mathcal{A})$, e o custo reduzido de um trajecto qualquer, p, é possível saber também o respectivo custo, $c(p)$.

Lema 5.9 *Seja p um trajecto de s para t, e seja p_1 o trajecto mais curto, também entre s e t, na rede $(\mathcal{N}, \mathcal{A})$. Então $c(p) = c(p_1) + \bar{c}(p)$.*
Demonstração: Sejam $p = \langle v_1, a_1, v_2, \ldots, a_{\ell_p - 1}, v_{\ell_p} \rangle$ e p_1 dois trajectos de $\mathcal{P}$, sendo p_1 o de menor custo naquele conjunto. Por definição de custo de um trajecto, $c(p) = \sum_p c_{ij}$; logo, por definição de custo reduzido, $\bar{c}_{ij} = \pi_j^t - \pi_i^t + c_{ij}$, podemos deduzir sucessivamente:

$$\begin{aligned}
c(p) &= \textstyle\sum_p c_{ij} \\
&= \textstyle\sum_p (\pi_i^t - \pi_j^t + \bar{c}_{ij}) \\
&= \textstyle\sum_p \bar{c}_{ij} + \sum_p (\pi_i^t - \pi_j^t) \\
&= \bar{c}(p) + (\pi_{v_1}^t - \pi_{v_2}^t + \pi_{v_2}^t - \pi_{v_3}^t + \cdots + \pi_{v_{\ell_p - 1}}^t - \pi_{v_{\ell_p}}^t) \\
&= \bar{c}(p) + \pi_{v_1}^t - \pi_{v_{\ell_p}}^t \\
&= \bar{c}(p) + \pi_s^t - \pi_t^t.
\end{aligned}$$

Como $\pi_t^t = 0$, então $c(p) = \bar{c}(p) + \pi_s^t = \bar{c}(p) + c(p_1)$, como pretendíamos mostrar. □

O lema 5.10, onde é demonstrado que é indiferente enumerar trajectos utilizando o seu custo ou o seu custo reduzido, é consequência imediata do lema 5.9.

Lema 5.10 *Sejam p e q trajectos de s para t em $(\mathcal{N}, \mathcal{A})$; então $c(p) \leq c(q)$ se e só se $\bar{c}(p) \leq \bar{c}(q)$.*
Demonstração: Pelo lema 5.9,

$$\bar{c}(p) = c(p) - c(p_1)$$

e

$$\bar{c}(q) = c(q) - c(p_1),$$

onde p_1 é o trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$, sendo então imediato que $c(p) \leq c(q)$ se e só se $\bar{c}(p) \leq \bar{c}(q)$, como pretendíamos. □

A utilização dos custos reduzidos dos arcos em substituição dos seus custos usuais permitirá simplificar a forma de guardar os trajectos calculados. Assim, no início do algoritmo é determinada a árvore dos trajectos mais curtos de todos os nós para t e, de acordo com a árvore construída, o custo de cada arco da rede é substituído pelo respectivo custo reduzido. Em seguida é utilizado o raciocínio apresentado, segundo o qual cada trajecto p_k dá origem, a partir do nó v_i^k de p_k, com $\alpha_k < i < \ell_k$, a trajectos da forma

$$\mathrm{sub}_k(s, v_i^k) \diamond \langle v_i^k, j\rangle \diamond \mathcal{T}_t(j),$$

para todo o $j \in \mathcal{D}(v_i^k) - \{v_{i+1}^k\}$. Estes trajectos, obtidos a partir de p_k, têm custo não inferior ao custo de p_k, como é demonstrado no lema 5.11.

Lema 5.11 *Para $k \in \{1, \ldots, K\}$, seja p_k um dos trajectos de s para t retirados de P durante a execução do algoritmo de Eppstein, e sejam $t_1, \ldots, t_r$ os trajectos determinados pelo mesmo algoritmo a partir de p_k. Então, para todo o $1 \le i \le r$, $c(p_k) \le c(t_i)$.*

Demonstração: Para $k = 1$, p_1 é o trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$, pelo que pertence à árvore $\mathcal{T}_t$. Então, pelo lema 5.7, todos os arcos de p_1 têm custo reduzido nulo, pelo que $\bar{c}(p_1) = 0$.

Os trajectos calculados a partir de p_1 separam-se deste num nó x e são da forma

$$t_x = \mathrm{sub}_1(s, x) \diamond \langle x, j\rangle \diamond \mathcal{T}_t(j).$$

O trajecto $\mathcal{T}_t(j)$ pertence também a $\mathcal{T}_t$, tendo portanto custo reduzido nulo. Assim, t_x contém quando muito um arco que não está naquela árvore, o arco (x, j) com custo reduzido $\bar{c}_{xj} \ge 0$; logo $\bar{c}(p_1) \le \bar{c}(t_x)$ e consequentemente, pelo lema 5.10, $c(p_1) \le c(t_x)$.

Para $k > 1$, para cada nó x que se segue ao nó desvio de p_k, são calculados trajectos da forma

$$t_x = \mathrm{sub}_k(s, x) \diamond \langle x, j\rangle \diamond \mathcal{T}_t(j).$$

Como anteriormente, $\mathcal{T}_t(j)$ é um trajecto da árvore $\mathcal{T}_t$. Então os arcos de p_k que não pertencem a esta árvore são também arcos de t_x. Além disso (x, j), que não está em p_k, pode pertencer ou não a $\mathcal{T}_t$, logo

$$\bar{c}(t_x) = \bar{c}(p_k) + \bar{c}_{xj} + \bar{c}(\mathcal{T}_t(j)) = \bar{c}(p_k) + \bar{c}_{xj}.$$

Como $\bar{c}_{xj} \ge 0$, então $\bar{c}(p_k) \le \bar{c}(t_x)$, e portanto, $c(p_k) \le c(t_x)$, novamente pelo lema 5.10, como pretendíamos concluir. $\square$

De acordo com o raciocínio descrito, cada trajecto p, obtido a partir de p_k, pode ser identificado por p_k, pelo nó desvio e/ou pelo arco em que p se separa de p_k. Então, para conhecer o custo desse trajecto p, e considerando que $p = \mathrm{sub}_k(s, i) \diamond \langle i, j\rangle \diamond \mathcal{T}_t(j)$, devemos calcular

$$c(p) = c(\mathrm{sub}_k(s, i)) + c_{ij} + \pi_j^t.$$

Isto obrigaria a conhecer o custo de $\text{sub}_k(s,i)$, para vários nós i de p_k. Se o custo de cada arco c_{ij} fosse substituído pelo respectivo custo reduzido, $\bar{c}_{ij}$, então o custo de p obter-se-ia facilmente a partir de

$$c(p) = c(p_1) + \bar{c}(p) = \pi_s^t + \bar{c}(p).$$

Como p tem, quando muito, mais um arco não pertencente a $\mathcal{T}_t$ relativamente aos arcos de p_k, e uma vez que $\bar{c}(p)$ indica o acréscimo do custo do trajecto ao optar por utilizar os arcos acrescentados na obtenção de p, $\bar{c}(p) = \bar{c}(p_k) + \bar{c}_{ij}$, logo

$$c(p) = \pi_s^t + \bar{c}(p_k) + \bar{c}_{ij}.$$

Utilizando a mudança de variável indicada basta então conhecer o custo reduzido de cada arco da rede e o custo de cada um dos trajectos p_k.

Deste modo, no início deste algoritmo é determinada a árvore $\mathcal{T}_t$ e, de acordo com a árvore construída, são alterados os custos dos arcos de $(\mathcal{N}, \mathcal{A})$, passando cada um a ter o valor do respectivo custo reduzido. Como foi referido, isto permite simplificar a identificação de cada trajecto determinado.

O algoritmo de Eppstein utilizando a mudança de variável indicada é descrito no algoritmo 5.4.

Teorema 5.4 *Sejam $p_1, \dots, p_K$ os trajectos de s para t na rede $(\mathcal{N}, \mathcal{A})$ enumerados pelo algoritmo de Eppstein; então $c(p_1) \leq c(p_2) \leq \cdots \leq c(p_K)$.*

Demonstração: Seja $k = 1$; dado que p_1 é o trajecto mais curto em $\mathcal{P}$, então $c(p_1) \leq c(p)$ é válido para todo o trajecto p de s para t em $(\mathcal{N}, \mathcal{A})$, e portanto $c(p_1) \leq c(p_2)$.

Demonstremos agora que $c(p_k) \leq c(p_{k+1})$.

Admitamos que no passo k do algoritmo $p_{k+1} \in P$; então, como o trajecto escolhido foi p_k, verifica-se que $c(p_k) \leq c(p_{k+1})$.

Se no passo k do algoritmo $p_{k+1} \notin P$, p_{k+1} foi determinado a partir de p_k; logo, pelo lema 5.11, $c(p_k) \leq c(p_{k+1})$, como pretendíamos. $\square$

Para ilustrar o funcionamento do algoritmo de Eppstein vamos simular apenas três dos seus passos, quando é aplicado à determinação dos $K = 3$ trajectos mais curtos de $s = 1$ para $t = 6$ na rede da figura 13.

O algoritmo inicia-se com a determinação da árvore dos trajectos mais curtos de todos os nós para t, representada anteriormente na figura 16.

Uma vez determinada esta árvore, inicia-se o cálculo dos trajectos mais curtos de 1 para 6. O primeiro trajecto calculado é $p_1 = \langle 1, 4, 7, 6 \rangle$, donde resulta $P = \{p_1\}$.

Algoritmo 5.4: *Algoritmo de Eppstein*

$\mathcal{T}_t \longleftarrow$ árvore dos trajectos mais curtos de todos os nós para t em $(\mathcal{N}, \mathcal{A})$;
`Para` (todo o $(i,j) \in \mathcal{A}$) `Fazer` $\bar{c}_{ij} \longleftarrow \pi_j^t - \pi_i^t + c_{ij}$;
$p \longleftarrow \mathcal{T}_t(s)$;
$P \longleftarrow \{p\}$;
$k \longleftarrow 1$;
`Enquanto` $(k \leq K)$ e $(P \neq \emptyset)$ `Fazer`
  $p_k \longleftarrow$ trajecto de P tal que $\bar{c}(p_k) \leq \bar{c}(p)$, qualquer que seja $p \in P$;
  $P \longleftarrow P - \{p_k\}$;
  `Se` $(k = 1)$ `Então` $\alpha_k \longleftarrow 0$;
  `Senão` $\alpha_k \longleftarrow$ ordem do nó desvio de p_k;
  `FimSe`
  `Para` (todo o $i \in \{\alpha_k + 1, \ldots, \ell_k - 1\}$) `Fazer`
    `Para` (todo o $j \in \mathcal{D}(v_i^k) - \{v_{i+1}^k\}$) `Fazer`
      $q_i^k \longleftarrow \langle v_i^k, j\rangle \diamond \mathcal{T}_t(j)$;
      $p_i^k \longleftarrow \mathrm{sub}_k(s, v_i^k) \diamond q_i^k$;
      $P \longleftarrow P \cup \{p_i^k\}$;
    `FimPara`
  `FimPara`
  $k \longleftarrow k + 1$;
`FimEnquanto`

No primeiro passo deste algoritmo, p_1 é retirado do conjunto P e a partir dele são calculados alguns trajectos. A árvore da figura 21 representa os trajectos calculados de 1 para 6 em $(\mathcal{N}, \mathcal{A})$, no final do primeiro passo deste algoritmo.

Ao ser executado o segundo passo, é determinado o trajecto $p_2 = \langle 1, 4, 6\rangle$. No entanto, como não é possível obter trajectos a partir de p_2, a árvore construída permanece inalterada. Assim, a árvore obtida após a a conclusão do terceiro passo coincide com a obtida após a a conclusão do terceiro passo utilizando a generalização do algoritmo de Yen, representada anteriormente na figura 18.

Terminamos aqui esta simulação que poderia continuar durante o número de passos pretendido, de acordo com o algoritmo 5.4.

O pior dos casos para a execução do algoritmo de Eppstein ocorre, como para o algoritmo GY, para uma rede do tipo da figura 19, em que todo o trajecto mais curto passa por todos os nós da rede e onde, além disso existe um arco de retorno, com custo nulo.

Para cada nó do subtrajecto de p_k seguinte ao nó desvio é necessário determinar

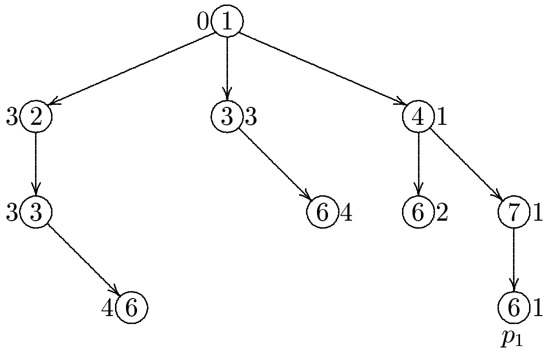

Figura 21

todos os trajectos de s para t que se desviem nesse nó. Ora, existem $m-n$ arcos da rede que não pertencem a p_k; isto é, dado um trajecto p_k são calculados $m-n$ novos trajectos. Assim, podemos afirmar que, no pior dos casos, o algoritmo de Eppstein tem uma complexidade de $\mathcal{O}(K(m-n))$.

5.2.5 Algoritmo MPS

No que se segue iremos descrever um algoritmo que enumera os K trajectos mais curtos entre dois nós, s e t, numa rede $(\mathcal{N}, \mathcal{A})$. Este algoritmo é devido a Martins, Pascoal e Santos e será por vezes designado, no que se segue, por algoritmo MPS.

Como foi referido no parágrafo anterior, no algoritmo de Eppstein, dado um trajecto p_k são calculados vários trajectos, a partir dos nós seguintes ao seu nó desvio. Em termos da construção da árvore dos trajectos isto significa que para cada um daqueles nós são acrescentados vários ramos à árvore referida. Dizemos por isso que a construção da árvore se realiza quase em "nível". Se relembrarmos que são calculados alguns dos trajectos constituídos pela concatenação de um arco e do trajecto mais curto desde o sucessor desse arco até t, que se separam de p_k em cada nó, é simples observar que facilmente este tipo de construção pode levar à determinação de um elevado número de trajectos p tais que $c(p) > c(p_K)$. A generalização do algoritmo de Yen tenta minimizar as hipóteses de determinação desses trajectos, calculando apenas um trajecto para cada nó seguinte ao nó desvio de p_k. No entanto isto só é conseguido com contrapartidas. De facto, a redução do número de trajectos calculados pela generalização do algoritmo de Yen é conseguida à custa da escolha de um nó j por forma a que $c_{ij} + c(\mathcal{T}_t(j))$ seja mínimo num certo conjunto.

Como foi referido, o algoritmo MPS tenta combinar a generalização do algoritmo de Yen e o algoritmo de Eppstein, tirando partido de uma ordenação especial do conjunto dos arcos da rede.

Em qualquer destes algoritmos são calculados sucessivos trajectos, que são guardados como elementos de um conjunto P, de candidatos ao $k^{\underline{\text{ésimo}}}$ trajecto mais curto, para um certo k, com $1 \leq k \leq K$. Em cada passo é escolhido em P o trajecto de menor custo ou de menor custo reduzido, conforme o algoritmo. O trajecto escolhido será p_k e a partir dele são calculados vários novos trajectos.

Relembremos sucintamente aqueles dois algoritmos.

Na generalização do algoritmo de Yen é calculado, para todo o nó v_i^k de p_k, com $\alpha_k \leq i \leq \ell_k - 1$, o trajecto mais curto que coincide com p_k de s até v_i^k, desviando-se nesse nó, e não estando o seu primeiro arco na árvore constituída por $\{p_1, \ldots, p_k\} \cup P$. O novo trajecto considerado é do tipo

$$p_i^k = \text{sub}_k(s, v_i^k) \diamond q_i^k,$$

onde $q_i^k = \langle v_i^k, j \rangle \diamond \mathcal{T}_t(j)$ é o trajecto de menor custo com aquela forma, e (v_i^k, j) é um arco que não pertence à árvore dos trajectos mais curtos de s para t nesse passo do algoritmo.

Como foi referido, calculando no início do algoritmo a árvore dos trajectos mais curtos de todos os nós para t, são conhecidos todos os trajectos $\mathcal{T}_t(j)$, com $j \in \mathcal{N} - \{t\}$. Para determinar um tal trajecto q_i^k, é necessário efectuar uma pesquisa, de modo a escolher de entre as combinações possíveis de arcos do tipo indicado e trajectos da forma $\mathcal{T}_t(j)$, aquela que tem menor custo.

Também no algoritmo de Eppstein, para cada nó v_i^k de p_k com $\alpha_k + 1 \leq i \leq \ell_k - 1$, são calculados todos os trajectos que coincidem com p_k de s até v_i^k e dele se desviam nesse nó . Entre os novos trajectos calculados encontra-se aquele que é determinado no algoritmo GY.

Além disso, no algoritmo de Eppstein, é realizada uma mudança de variável que facilita a identificação dos trajectos mais curtos determinados. Esta mudança de variável permite ainda obter uma certa ordenação do conjunto dos arcos da rede. Como iremos descrever em seguida, esta ordenação permite facilitar o cálculo dos trajectos pretendidos, simplificando a pesquisa efectuada pela generalização do algoritmo de Yen.

Pretendemos que os arcos que irão dar origem a trajectos com menor custo sejam os primeiros a ser escolhidos. Se relembramos que o custo reduzido de cada arco representa a distância a adicionar ao custo do trajecto mais curto quando aquele arco é utilizado no novo trajecto, parece natural pensar em ordenar o conjunto dos arcos da rede por ordem crescente dos seus custos reduzidos.

Seja i um nó de uma rede $(\mathcal{N}, \mathcal{A})$, e consideremos $\mathcal{F}(i) = \{(j, i) : (j, i) \in \mathcal{A}\}$

o conjunto dos arcos da rede que terminam no nó i; analogamente, consideremos ainda $\mathcal{I}(i) = \{(i,j) : (i,j) \in \mathcal{A}\}$ o conjunto dos arcos da rede que se iniciam em i. Denotemos por m_i o número de arcos que têm início em i, ou seja, o número de elementos de $\mathcal{I}(i)$. É de notar que o número total de arcos da rede é dado por $m = \sum_{i\in\mathcal{N}} m_i$. Para cada $i \in \mathcal{N}$, o conjunto $\mathcal{I}(i)$ pode ser ordenado da forma seguinte

$$\mathcal{I}(i) = \{(i, j_1^i), \ldots, (i, j_{m_i}^i)\},$$

onde $\bar{c}_{ij_u^i} \leq \bar{c}_{ij_{u+1}^i}$, para $1 \leq u \leq m_i - 1$.

Consideremos que o conjunto dos nós é dado por $\mathcal{N} = \{1, \ldots, n\}$. Iremos então organizar o conjunto dos arcos da seguinte forma

$$\mathcal{A} = \{(1, j_1^1), \ldots, (1, j_{m_1}^1), (2, j_1^2), \ldots, (2, j_{m_2}^2), \ldots, (n, j_1^n), \ldots, (n, j_{m_n}^n)\}.$$

Esta representação da rede permite obter os arcos que se iniciam num nó i por ordem não decrescente dos respectivos custos reduzidos. Organizar o conjunto $\mathcal{A}$ da forma indicada equivale ainda a impôr uma ordenação entre os arcos da rede, de acordo com as condições que se seguem. Dados dois arcos $(i,x), (j,y) \in \mathcal{A}$, diremos que (i,x) é anterior a (j,y) se $i < j$ ou $i = j$ e $\bar{c}_{ix} \leq \bar{c}_{jy}$.

Quando o conjunto dos arcos de uma rede está ordenado desta forma dizemos que está na *sorted forward star form*; para mais pormenores acerca desta estrutura de dados consultar [6].

Com esta ordenação dos arcos da rede torna-se possível a escolha imediata do arco que irá aumentar o menos possível o custo de um trajecto que o venha a conter.

Depois de ordenar a rede nesta forma é possível iniciar a determinação dos trajectos p_k, de acordo com o processo utilizado na generalização do algoritmo de Yen. A partir de p_k, e para cada v_i^k, com $\alpha_k \leq i \leq \ell_k - 1$, é calculado o trajecto

$$p_i^k = \mathrm{sub}_k(s, v_i^k) \diamond \langle v_i^k, j\rangle \diamond \mathcal{T}_t(j),$$

onde o arco (v_i^k, j) não deve pertencer à árvore dos trajectos mais curtos com raiz em s nesse passo do algoritmo. Assim, iremos escolher o arco de $\mathcal{I}(v_i^k)$ que se segue a a_i^k na rede ordenada como foi indicado. Além disso, se $a_i^k \equiv (v_i^k, j_u)$, então o arco escolhido deve ser (v_i^k, j_{u+1}), caso exista.

O algoritmo de Martins, Pascoal e Santos apresentado neste parágrafo é descrito de forma sumária no algoritmo 5.5.

Como é demonstrado no lema 5.12, neste algoritmo um trajecto origina sempre trajectos com uma distância não inferior à sua.

Algoritmo 5.5: *Algoritmo de Martins, Pascoal e Santos*

$\mathcal{T}_t \longleftarrow$ árvore dos trajectos mais curtos de todos os nós para t em $(\mathcal{N}, \mathcal{A})$;
`Para` (todo o $(i,j) \in \mathcal{A}$) `fazer` $\bar{c}_{ij} \longleftarrow \pi_j^t - \pi_i^t + c_{ij}$;
colocar $(\mathcal{N}, \mathcal{A})$ na *sorted forward star form*;
$p \longleftarrow \mathcal{T}_t(s)$;
$P \longleftarrow \{p\}$;
$k \longleftarrow 1$;
`Enquanto` $(k \leq K)$ e $(P \neq \emptyset)$ `Fazer`
  $p_k \longleftarrow$ trajecto de P tal que $\bar{c}(p_k) \leq \bar{c}(p)$, qualquer que seja $p \in P$;
  $P \longleftarrow P - \{p_k\}$;
  $\alpha_k \longleftarrow$ ordem do nó desvio de p_k;
  `Para` (todo o $i \in \{\alpha_k, \ldots, \ell_k - 1\}$) `Fazer`
    $j \longleftarrow$ nó de $\mathcal{D}(v_i^k)$ tal que (v_i^k, j) é o primeiro arco da rede seguinte a a_i^k;
    `Se` (j está definido) `Então`
      $q_i^k \longleftarrow \langle v_i^k, j \rangle \diamond \mathcal{T}_t(j)$;
      $p_i^k \longleftarrow \mathrm{sub}_k(s, v_i^k) \diamond q_i^k$;
      $P \longleftarrow P \cup \{p_i^k\}$;
    `FimSe`
  `FimPara`
  $k \longleftarrow k + 1$;
`FimEnquanto`

Lema 5.12 *Dado $k \in \{1, \ldots, K\}$, seja p_k um dos trajectos de s para t retirado de P durante a execução do algoritmo MPS, e sejam $t_1, \ldots, t_r$ os trajectos determinados pelo mesmo algoritmo a partir de p_k. Então, para todo o $i \in \{1, \ldots, r\}$, $c(p_k) \leq c(t_i)$.*

Demonstração: Seja $k = 1$; p_1 é o trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$, e então, pelo lema 5.7, $\bar{c}(p_1) = 0$.

Os trajectos calculados a partir de p_1 separam-se deste num nó x e são da forma

$$t_x = \mathrm{sub}_1(s, x) \diamond \langle x, j \rangle \diamond \mathcal{T}_t(j).$$

Se $x = v_i^1$, então (x, j) é o arco da rede ordenada que se segue a a_i^1. Então $\bar{c}_{xj} \geq \bar{c}_{a_i^1} = 0$ e $\bar{c}(\mathcal{T}_t(j)) = 0$, donde $\bar{c}(p_1) \leq \bar{c}(t_x) = \bar{c}_{xj}$.

Seja $k > 1$; para todo o nó x de p_k, tal que x coincide com o nó desvio de p_k ou é um dos nós intermédios do subtrajecto de p_k que se define do nó desvio para t, são calculados trajectos da forma

$$t_x = \mathrm{sub}_k(s, x) \diamond \langle x, j \rangle \diamond \mathcal{T}_t(j).$$

Se $x = v_i^k$, (x, j) é o arco da rede ordenada que se segue a a_i^k. Então, mais uma vez, pela forma como foi escolhido, (x, j) tem custo reduzido não inferior a a_i^k.

Os arcos de p_k que não pertencem à árvore $\mathcal{T}_t$, caso existam, são arcos anteriores ao nó desvio e eventualmente ao arco $a_{\alpha_k}^k$, no conjunto dos arcos da rede depois de ordenado. Analogamente, os arcos de t_x que não estão naquela árvore, coincidem com os de p_k que estão nas mesmas condições e são anteriores a x e, eventualmente, o arco (x, j).

Consideremos que o nó $x = v_i^k$ de p_k é seguinte ao nó desvio, isto é, $\alpha_k < i < l_k$. Então $\bar{c}(t_x) = \bar{c}(p_k) + \bar{c}_{xj} \geq \bar{c}(p_k)$, uma vez que $\bar{c}_{uv} \geq 0$ para todo o arco (u, v) da rede $(\mathcal{N}, \mathcal{A})$.

Consideremos agora que o nó $x = v_i^k$ de p_k coincide com o nó desvio, ou seja, $i = \alpha_k$. Então $\bar{c}(t_x) = \bar{c}(p_k) - \bar{c}_{a_{\alpha_k}^k} + \bar{c}_{xj} \geq \bar{c}(p_k)$, uma vez que, como vimos, $\bar{c}_{xj} \geq \bar{c}_{a_i^k}$, e portanto $\bar{c}_{xj} - \bar{c}_{a_i^k} \geq 0$. □

Este lema pode ser utilizado para demonstrar o teorema 5.5, enunciado em seguida, que garante que os trajectos $p_1, \ldots, p_K$ são determinados neste algoritmo por ordem não decrescente de custos. A demonstração deste teorema é análoga à do teorema 5.4, razão pela qual não é apresentada novamente.

Teorema 5.5 *Sejam $p_1, \ldots, p_K$ os trajectos de s para t em $(\mathcal{N}, \mathcal{A})$ determinados pelo algoritmo MPS; então $c(p_1) \leq c(p_2) \leq \cdots \leq c(p_K)$.*

Embora o algoritmo MPS aqui apresentado exija a ordenação da rede na forma que foi descrita, esta desvantagem é compensada pela elevada eficiência do algoritmo.

Tal como acontecia nos algoritmos GY e de Eppstein, o pior dos casos para a execução do algoritmo MPS ocorre para uma rede do tipo representado na figura 19, onde o trajecto mais curto é constituído por todos os nós da rede e onde existe um arco de retorno de custo nulo. Assim, para cada p_k, com $k \in \{1, \ldots, K\}$, são determinados, quando muito, n novos trajectos, correspondendo cada um deles a um dos nós de p_k seguintes ao nó desvio. Então, no pior dos casos, o algoritmo MPS tem uma complexidade de $\mathcal{O}(Kn)$. Estes cálculos foram realizados supondo que a rede foi dada na forma *sorted forward star form*. De outro modo seria necessário necessário ordenar a rede naquela forma, o que pode ser conseguido com duas ou três passagens pelo conjunto dos arcos da rede, o que significa que efectua um número de operações da $\mathcal{O}(m)$. No total o algoritmo MPS tem então uma ordem de complexidade de $\mathcal{O}(Kn + m)$.

Para concluir esta subsecção, é de notar que os algoritmos aqui apresentados, exceptuando o algoritmo de Eppstein, são originais. Todos eles, como foi referido, têm por base a construção da árvore dos trajectos mais curtos de s para t, tentando começar por "colocar" naquela árvore os ramos correspondentes a trajectos com menor custo. A forma como cada um destes algoritmos consegue realizar esta tarefa determina a sua maior ou menor eficiência. Tentámos apresentar estes algoritmos por ordem decrescente de simplicidade e crescente de eficiência.

É ainda de salientar que apesar de serem utilizadas apenas redes orientadas na descrição aqui realizada, estes algoritmos são válidos também no caso de existirem arcos não orientados uma vez que, como foi referido na secção 1, cada um destes pode ser visto como dois arcos orientados.

5.3 Algoritmos Baseados no Princípio de Optimalidade

Na subsecção anterior foram descritos vários algoritmos para a resolução do problema dos K Trajectos Mais Curtos, todos eles tendo por base a construção de uma árvore de trajectos de s para t numa rede $(\mathcal{N}, \mathcal{A})$, contendo os K mais curtos. Como foi referido, alguns algoritmos para o problema dos K Trajectos Mais Curtos podem ser desenvolvidos, agora com base num princípio que este problema verifica. De facto, de modo semelhante ao que acontecia para $K = 1$ e foi referido na secção 3, para $K > 1$ o problema em estudo verifica o **Princípio de Optimalidade** enunciado em seguida no lema 5.13.

Lema 5.13 *O $k^{\underline{\text{ésimo}}}$ trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$ é constituído por subtrajectos que são $r^{\underline{\text{ésimo}}}$ mais curtos, com $1 \leq r \leq k$.*

Demonstração: Sejam $p_1, \ldots, p_{k-1}$ os $k-1$ trajectos mais curtos de s para t em $(\mathcal{N}, \mathcal{A})$. Consideremos ainda que $p^\star \in \mathcal{P}$ é $k^{\underline{\text{ésimo}}}$ trajecto mais curto, com $k \in \{1, \ldots, K\}$. Então $c(p^\star) \leq c(p)$, para todo o $p \in \mathcal{P} - \{p_1, \ldots, p_{k-1}\}$.

Sejam u e v, nós distintos de $p^\star$; pretendemos mostrar que $q = \text{sub}_{p^\star}(u, v)$ é um trajecto $r^{\underline{\text{ésimo}}}$ mais curto de u para v, com $1 \leq r \leq k$.

Suponhamos então que existem k trajectos distintos de u para v, $q_1, \ldots, q_k$, tais que $c(q_i) \leq c(q_{i+1})$, para $i \in \{1, \ldots, k-1\}$, e $c(q_i) \leq c(p)$, para todo o $i \in \{1, \ldots, k\}$ e $p \in \mathcal{P}_{uv} - \{q_1, \ldots, q_k\}$, mais curtos do que q, ou seja, $c(q_i) < c(q)$.

Então, para cada $i \in \{1, \ldots, k\}$, $\text{sub}_{p^\star}(s, u) \diamond q_i \diamond \text{sub}_{p^\star}(v, t)$ é um trajecto de s para t e

$$\begin{aligned} c(\text{sub}_{p^\star}(s, u) \diamond q_i \diamond \text{sub}_{p^\star}(v, t)) &= c(\text{sub}_{p^\star}(s, u)) + c(q_i) + c(\text{sub}_{p^\star}(v, t)) \\ &< c(\text{sub}_{p^\star}(s, u)) + c(q) + c(\text{sub}_{p^\star}(v, t)) \\ &= c(p^\star), \end{aligned}$$

pelo que $\text{sub}_{p^\star}(s, u) \diamond q_1 \diamond \text{sub}_{p^\star}(v, t), \ldots, \text{sub}_{p^\star}(s, u) \diamond q_k \diamond \text{sub}_{p^\star}(v, t)$ são k trajectos mais curtos em $(\mathcal{N}, \mathcal{A})$, e $p^\star$ não é o $k^{\underline{\text{ésimo}}}$ trajecto mais curto, como foi suposto. $\square$

De forma análoga à realizada para o problema do Trajecto Mais Curto, podemos desenvolver algoritmos para resolução do problema dos K Trajectos Mais Curtos baseados neste princípio.

Nesta subsecção iremos descrever apenas um algoritmo baseado neste princípio.

5.3.1 Algoritmo MS

Como referimos podem ser desenvolvidos algoritmos baseados no Princípio de Optimalidade para o problema dos K Trajectos Mais Curtos. Neste parágrafo iremos descrever as várias versões de um algoritmo para a resolução deste problema e que se baseia no princípio anteriormente enunciado.

A primeira versão deste algoritmo deve-se a Martins, [14], versão esta que tem sido sujeita a sucessivos melhoramentos, [15]. A versão mais recente deve-se a Martins e Santos, [17], e é conhecida por algoritmo MS.

A ideia principal deste algoritmo consiste no apagamento de trajectos de uma rede, e na determinação de novos trajectos após cada remoção.

Seja p_1 o trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$, trajecto este que pode ser determinado por um qualquer dos algoritmos anteriormente indicados para este problema.

Uma vez determinado p_1, pretendemos em seguida determinar p_2, o segundo trajecto mais curto na mesma rede. Este problema pode ser reduzido ao anterior se a rede inicial for transformada numa outra, onde é possível determinar todos os trajectos de $(\mathcal{N}, \mathcal{A})$, excepto p_1. Bastaria então recorrer novamente ao mesmo algoritmo e calcular o trajecto mais curto na rede transformada, dado que, se não é possível definir o trajecto p_1, então o trajecto mais curto nessa rede corresponderá ao segundo mais curto em $(\mathcal{N}, \mathcal{A})$.

A questão que se coloca em seguida é a de saber transformar uma dada rede, por forma a obter uma outra, excluindo apenas um dado trajecto; ou seja, para $k \in \{1, \ldots, K\}$, dados uma rede $(\mathcal{N}_k, \mathcal{A}_k)$ e um trajecto p_k em $(\mathcal{N}_k, \mathcal{A}_k)$, como obter outra rede $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$, onde podemos determinar todos os trajectos de $(\mathcal{N}_k, \mathcal{A}_k)$, excepto p_k. Admitindo que é possível transformar $(\mathcal{N}_k, \mathcal{A}_k)$ em $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$, "apagando" um dado trajecto p_k de $(\mathcal{N}_k, \mathcal{A}_k)$, resulta o algoritmo para enumerar os K trajectos mais curtos, descrito no algoritmo 5.6. Uma vez conhecido o procedimento que permite apagar p_k de $(\mathcal{N}_k, \mathcal{A}_k)$, para $k \in \{1, \ldots, K-1\}$, será descrito o mesmo algoritmo de forma mais pormenorizada.

Por forma a permitir a repetição, quer do nó inicial, s, quer do terminal, t, num trajecto, a rede dada $(\mathcal{N}, \mathcal{A})$ pode ser acrescentada, como foi referido anteriormente, com os nós, S e T, e com os arcos (S, s) e (t, T), ambos de custo nulo. Os trajectos

Algoritmo 5.6: *Algoritmo de Martins*

$(\mathcal{N}_1, \mathcal{A}_1) \longleftarrow (\mathcal{N}, \mathcal{A})$;
$p_1 \longleftarrow$ trajecto mais curto de s para t em $(\mathcal{N}_1, \mathcal{A}_1)$;
`Para` (todo o $k \in \{2, \ldots, K\}$) `Fazer`
  $(\mathcal{N}_k, \mathcal{A}_k) \longleftarrow$ transformada de $(\mathcal{N}_{k-1}, \mathcal{A}_{k-1})$ de onde é apagado p_{k-1};
  $p_k \longleftarrow$ trajecto mais curto de s para t em $(\mathcal{N}_k, \mathcal{A}_k)$;
`FimPara`

passam a ser determinados de S para T. É de notar que qualquer trajecto tem pelo menos três arcos.

A fim de facilitar a descrição deste algoritmo, admitimos que a rede $(\mathcal{N}, \mathcal{A})$ foi acrescentada, assumindo pois que qualquer trajecto é definido de s para t e é constituído por, pelo menos, três arcos.

Pretendemos pois transformar $(\mathcal{N}, \mathcal{A})$ numa outra rede $(\mathcal{N}', \mathcal{A}')$, onde é possível realizar todos os trajectos de s para t que se definem em $(\mathcal{N}, \mathcal{A})$, excepto um trajecto dado, p.

Consideremos então

$$p = \langle v_1, a_1, v_2, a_2, \ldots, v_{\ell_p - 1}, a_{\ell_p - 1}, v_{\ell_p} \rangle,$$

onde $\ell_p \geq 3$, um trajecto de s para t em $(\mathcal{N}, \mathcal{A})$.

O conjunto dos nós $\mathcal{N}'$ será constituído pelo conjunto dos nós da rede original, ao qual são acrescentadas "cópias" dos nós do trajecto p que não o inicial e o terminal, por forma a obter alternativa aos subtrajectos de p, que é possível definir de s para v_i, para $i \in \{1, \ldots, \ell_p\}$.

Como veremos, o conjunto $\mathcal{A}'$ é constituído não só por elementos de $\mathcal{A}$, mas também por alguns novos arcos que ligam nós da rede original às "cópias" dos nós de p e por arcos que ligam as "cópias" entre si.

É de notar que um trajecto q' de $(\mathcal{N}', \mathcal{A}')$ pode conter nós e arcos que não pertençam a $(\mathcal{N}, \mathcal{A})$; no entanto, estes nós e arcos constituem uma cópia de nós e arcos de $(\mathcal{N}, \mathcal{A})$. Diremos nesse caso que $q' = q$, onde q é o trajecto de $(\mathcal{N}, \mathcal{A})$ que resulta de q' substituindo os nós e arcos de $(\mathcal{N}', \mathcal{A}')$ não existentes em $(\mathcal{N}, \mathcal{A})$ pelos respectivos nós e arcos desta última rede. É neste sentido que afirmamos que os trajectos de s para t em $(\mathcal{N}', \mathcal{A}')$ são os trajectos de s para t em $(\mathcal{N}, \mathcal{A})$. Diremos neste caso que q' é o **trajecto alternativo** ou **alternativa** de q.

Seja p' o trajecto de s para t que é alternativa do trajecto p a ser removido de $(\mathcal{N}, \mathcal{A})$. Se construíssemos $(\mathcal{N}', \mathcal{A}')$ de modo a que contivesse p e uma sua "cópia", continuaria a ser possível determinar aquele trajecto na nova rede. Para que isto não aconteça devem ser retirados da rede o primeiro arco de p' e o último de p, ou o último arco de p' e o primeiro de p. No que se segue, a rede transformada, $(\mathcal{N}', \mathcal{A}')$ é a que resulta da consideração do primeiro caso.

Lembremos que, dado um nó i de uma rede, $\mathcal{F}(i)$ designa o conjunto dos arcos da rede que terminam em i. Denotemos ainda por $\mathcal{F}'(i)$ o conjunto análogo, na rede $(\mathcal{N}', \mathcal{A}')$.

Assim, dada a rede $(\mathcal{N}, \mathcal{A})$ e o trajecto $p = \langle v_1, \ldots, v_{\ell_p} \rangle$, de $(\mathcal{N}, \mathcal{A})$, a rede transformada, $(\mathcal{N}', \mathcal{A}')$, é definida por:

1. $\mathcal{N}' = \mathcal{N} \cup \{v'_2, \ldots, v'_{\ell_p - 1}\}$;

2. $\mathcal{A}' = (\mathcal{A} - \{(v_{\ell_p - 1}, t)\}) \cup \mathcal{F}'(v'_2) \cup \ldots \cup \mathcal{F}'(v'_{\ell_p - 1}) \cup \{(v'_{\ell_p - 1}, t)\}$, onde:

   (a) $\mathcal{F}'(v'_2) = \{(u, v'_2) : (u, v_2) \in \mathcal{F}(v_2) - \{(s, v_2)\}\}$;

   (b) $\mathcal{F}'(v'_i) = \{(u, v'_i) : (u, v_i) \in \mathcal{F}(v_i) - \{(v_{i-1}, v_i)\}\} \cup \{(v'_{i-1}, v'_i)\}$, para todo o $2 < i < \ell_p$.

Resta-nos identificar o custo a associar a cada um dos arcos da nova rede.

Pretendemos que cada nó acrescentado constitua uma "cópia" do nó correspondente na rede inicial, por forma a que todos os trajectos, com excepção de p, se continuem a poder definir na rede transformada, em condições idênticas. Concluímos pois que os trajectos alternativos na nova rede deverão ter custo igual aos seus correspondentes na rede original. Para tal os arcos da rede original mantêm o seu custo e os custos dos novos arcos dependem dos custos dos arcos iniciais que lhes deram origem. Teremos então:

- $c_{i'j'} = c_{ij}$, para $(i', j') \in \mathcal{A}'$, quaisquer que sejam $i', j' \in \mathcal{N}' - \mathcal{N}$;

- $c_{ij'} = c_{ij}$, para $(i, j') \in \mathcal{A}'$, quaisquer que sejam $i \in \mathcal{N}$ e $j' \in \mathcal{N}' - \mathcal{N}$;

- $c_{i'j} = c_{ij}$, para $(i', j) \in \mathcal{A}'$, quaisquer que sejam $j \in \mathcal{N}$ e $i' \in \mathcal{N}' - \mathcal{N}$.

Mostramos, no teorema 5.6, que os trajectos diferentes de p que se podem realizar em $(\mathcal{N}, \mathcal{A})$, se podem realizar também em $(\mathcal{N}', \mathcal{A}')$, e vice-versa.

Teorema 5.6 *Seja $p = \langle v_1, \ldots, v_{\ell_p} \rangle$ um trajecto de s para t numa dada rede $(\mathcal{N}, \mathcal{A})$ e seja $(\mathcal{N}', \mathcal{A}')$ a rede obtida de $(\mathcal{N}, \mathcal{A})$ tal que:*

1. $\mathcal{N}' = \mathcal{N} \cup \{v'_2, \ldots, v'_{\ell_p - 1}\}$;

2. $\mathcal{A}' = (\mathcal{A} - \{(v_{\ell_p-1}, t)\}) \cup \mathcal{F}'(v_2') \cup \ldots \cup \mathcal{F}'(v_{\ell_p-1}') \cup \{(v_{\ell_p-1}', t)\}$*, onde:*

   (a) $\mathcal{F}'(v_2') = \{(u, v_2') : (u, v_2) \in \mathcal{F}(v_2) - \{(s, v_2)\}\}$*;*

   (b) $\mathcal{F}'(v_i') = \{(u, v_i') : (u, v_i) \in \mathcal{F}(v_i) - \{(v_{i-1}, v_i)\}\} \cup \{(v_{i-1}', v_i')\}$*, para todo o* $2 < i < \ell_p$*.*

Designando por $\mathcal{P}'$ *o conjunto dos trajectos de* s *para* t *na rede* $(\mathcal{N}', \mathcal{A}')$*, então* $\mathcal{P} - \{p\} = \mathcal{P}'$*.*

Demonstração: Seja $p = \langle v_1, \ldots, v_{\ell_p}\rangle$, com $v_1 = s$ e $v_{\ell_p} = t$, um qualquer trajecto de s para t numa rede $(\mathcal{N}, \mathcal{A})$.

Comecemos por demonstrar que $\mathcal{P} - \{p\} \subseteq \mathcal{P}'$.

Seja $q = \langle n_1, \ldots, n_{\ell_q}\rangle$ um trajecto de $\mathcal{P} - \{p\}$.

Uma vez que $q \neq p$, existe pelo menos um valor j, tal que $1 < j < \ell_p$ e $1 < j < \ell_q$, para o qual $v_j \neq n_j$.

Pela construção de $(\mathcal{N}', \mathcal{A}')$, todos os arcos da rede inicial, com a excepção de (v_{ℓ_p-1}, t), continuam a ser arcos da nova rede. Assim, se q não contiver aquele arco é imediato que q é um trajecto em $\mathcal{P}'$ de s para t em $(\mathcal{N}', \mathcal{A}')$, ou seja, pertencente a $\mathcal{P}'$.

Suponhamos, sem perda de generalidade, que aquele é o último arco de q, o que é possível dado que, acrescentando um super-nó terminal à rede, existe um único arco que termina em T. Então $n_{\ell_q-1} = v_{\ell_p-1}$.

Seja j o menor inteiro tal que $n_{\ell_q-j} \neq v_{\ell_p-j}$. Como q e p são diferentes mas o seu último arco é o mesmo, então $2 \leq j \leq \min\{\ell_q, \ell_p\} - 2$. Assim, podemos definir o trajecto $\mathrm{sub}_q(s, n_{\ell_q-j})$ na rede $(\mathcal{N}', \mathcal{A}')$, mas, no entanto, não podemos definir $\mathrm{sub}_q(n_{\ell_q-j+1}, t) = \mathrm{sub}_p(v_{\ell_p-j+1}, t)$, uma vez que o seu último arco não pertence à nova rede. No entanto, pela forma como foi obtida $(\mathcal{N}', \mathcal{A}')$, quer o nó $n_{\ell_q-j+1}' = v_{\ell_p-j+1}'$ quer o arco $(n_{\ell_q-j}, n_{\ell_q-j+1}')$ pertencem à nova rede. Assim, o trajecto $\mathrm{sub}_q(n_{\ell_q-j}, t)$ é substituído por

$$\langle n_{\ell_q-j}, n_{\ell_q-j+1}', n_{\ell_q-j+2}', \ldots, n_{\ell_q-1}', t\rangle = \langle v_{\ell_p-j}, v_{\ell_p-j+1}', v_{\ell_p-j+2}', \ldots, v_{\ell_p-1}', t\rangle,$$

pelo que o trajecto de s para t em $(\mathcal{N}', \mathcal{A}')$,

$$q' = \ \mathrm{sub}_q(s, n_{\ell_q-j}) \diamond \langle n_{\ell_q-j}, n_{\ell_q-j+1}', \ldots, n_{\ell_q-1}', t\rangle,$$

é escolhido como alternativo de q e podemos dizer que $q \in \mathcal{P}'$.

Demonstremos agora que $\mathcal{P}' \subseteq \mathcal{P} - \{p\}$.

Seja $q' = \langle n_1, \ldots, n_{\ell_{q'}}\rangle$ um trajecto de $\mathcal{P}'$.

Mostremos que $q' \neq p$, começando por supor o contrário, isto é, supondo que $q' = p$.

Ora $(n_{\ell_{q'}-1}, t) \neq (v_{\ell_p-1}, t)$ uma vez que (v_{ℓ_p-1}, t) não é um arco da nova rede; então $n_{\ell_{q'}-1}$ teria de ser uma alternativa para v_{ℓ_p-1}, isto é, $n_{\ell_{q'}-1} = v'_{\ell_p-1}$. Como os arcos (v_{i-1}, v'_i), para $2 < i < \ell_p$, também não pertencem à nova rede, para q' ser igual a p, os nós $n_2, \ldots, n_{\ell_{q'}-1}$ teriam de ser alternativas a $v_2, \ldots, v_{\ell_p-1}$. Mas então, como (s, v'_2) não pertence à nova rede, concluímos que q' não é alternativa a p, isto é, $q' \neq p$, como pretendíamos.

Uma vez demonstrado que $q' \neq p$, mostremos agora que $q' \in \mathcal{P}$, o que significa que $q' \in \mathcal{P} - \{p\}$.

Para isso, atendendo à construção de $(\mathcal{N}', \mathcal{A}')$, basta substituir os arcos do tipo (v'_{i-1}, v'_i) por (v_{i-1}, v_i), para $2 < i < \ell_p$, o arco (v'_{ℓ_p-1}, t) por (v_{ℓ_p-1}, t), e finalmente, os arcos do tipo (u, v'_i) por (u, v_i), para $(u, v'_i) \in \mathcal{F}(v_i)$ e $2 < i < \ell_p$, ficando assim concluída a demonstração. □

Como consequência do teorema 5.6 podemos obter o corolário 5.6.1, apresentado em seguida.

Corolario 5.6.1 *Trajectos alternativos em $(\mathcal{N}, \mathcal{A})$ e em $(\mathcal{N}', \mathcal{A}')$ têm custos iguais.*
Demonstração: Seja q' um trajecto alternativo, em $(\mathcal{N}', \mathcal{A}')$, ao trajecto q, em $(\mathcal{N}, \mathcal{A})$. Basta observar, pelo teorema 5.6, que q e q' utilizam os mesmos arcos ou arcos alternativos. Pela forma como estão definidos os custos dos arcos da nova rede estes têm o mesmo custo que os correspondentes na rede original, pelo que fica demonstrado que $c(q) = c(q')$, como pretendíamos. □

Por forma a ilustrar a construção desta nova rede, $(\mathcal{N}', \mathcal{A}')$, voltemos a considerar a rede da figura 5. Após terem sido acrescentados os super-nós inicial e terminal obtemos a rede da figura 8, onde o trajecto mais curto é $p = p_1 = \langle S, 1, 2, 3, 5, T\rangle$.

Consideremos que a rede inicial é a rede aumentada, que iremos denotar por $(\mathcal{N}, \mathcal{A})$. A rede representada na figura 22 é $(\mathcal{N}', \mathcal{A}')$, obtida como foi descrito anteriormente a partir de $(\mathcal{N}, \mathcal{A})$ e de p.

É de notar que na nova rede $(\mathcal{N}', \mathcal{A}')$, não é possível definir o trajecto mais curto em $(\mathcal{N}, \mathcal{A})$, quer utilizando os nós de $(\mathcal{N}, \mathcal{A})$, quer utilizando também as "cópias" daqueles nós que foram acrescentadas, uma vez que não existem os arcos $(S, 1')$ e $(5, T)$.

Apesar disso, é possível definir qualquer outro trajecto de S para T, tendo em conta que cada nó i' é uma "cópia" de i. Por exemplo, ao trajecto $\langle S, 1, 3, 5, T\rangle$ em $(\mathcal{N}, \mathcal{A})$ corresponde o trajecto $\langle S, 1, 3', 5', T\rangle$ em $(\mathcal{N}', \mathcal{A}')$. Este último é o mais curto na nova rede e tem custo 4, exactamente o mesmo que a sua alternativa em $(\mathcal{N}, \mathcal{A})$. Assim, e como referimos anteriormente, $p_2 = \langle S, 1, 3, 5, T\rangle$ é o segundo trajecto mais curto de S para T na rede original.

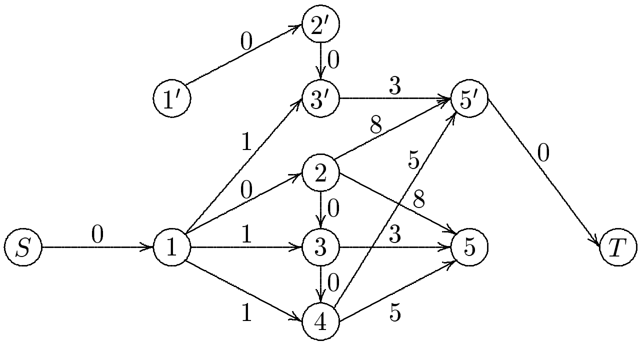

Figura 22

No algoritmo de Martins, que aqui descrevemos, é construída uma sequência de redes, $\{(\mathcal{N}_1, \mathcal{A}_1), \ldots, (\mathcal{N}_K, \mathcal{A}_K)\}$, tal que $(\mathcal{N}_1, \mathcal{A}_1) = (\mathcal{N}, \mathcal{A})$ e, para todo o $k \geq 2$, $(\mathcal{N}_k, \mathcal{A}_k) = (\mathcal{N}'_{k-1}, \mathcal{A}'_{k-1})$. No teorema 5.7 é demonstrado que o trajecto mais curto em $(\mathcal{N}_k, \mathcal{A}_k)$ é o $k^{\underline{\text{ésimo}}}$ trajecto mais curto em $(\mathcal{N}, \mathcal{A})$, para $k \geq 1$.

Teorema 5.7 *O trajecto mais curto de s para t em $(\mathcal{N}_k, \mathcal{A}_k)$ é o $k^{\underline{\text{ésimo}}}$ mais curto de s para t em $(\mathcal{N}, \mathcal{A})$, para $1 \leq k \leq K$.*

Demonstração: Por definição, $(\mathcal{N}_1, \mathcal{A}_1) = (\mathcal{N}, \mathcal{A})$, logo p_1 é o trajecto mais curto naquela rede.

Suponhamos que sendo $p^\star$ o trajecto mais curto em $(\mathcal{N}_k, \mathcal{A}_k)$, $p^\star$ é o $k^{\underline{\text{ésimo}}}$ mais curto em $(\mathcal{N}, \mathcal{A})$, isto é, $p^\star = p_k$.

Ora, pela construção destas redes, $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1}) = (\mathcal{N}'_k, \mathcal{A}'_k)$. Como vimos no teorema 5.6, os trajectos de $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$ coincidem com os trajectos de $(\mathcal{N}_k, \mathcal{A}_k)$, excluindo p_k, que não se pode definir em $(\mathcal{N}_k, \mathcal{A}_k)$.

Designemos por $p^\star$ o trajecto mais curto em $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$. Então $p^\star$ é tal que $c(p^\star) \leq c(p)$, para todo o p definido em $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$; ou seja, para todo o p definido em $(\mathcal{N}_k, \mathcal{A}_k)$, excepto p_k. Como p_k é o mais curto em $(\mathcal{N}_k, \mathcal{A}_k)$, teremos ainda que $c(p_k) \leq c(p^\star)$. Nesse caso, $p^\star$ é o segundo trajecto mais curto em $(\mathcal{N}_k, \mathcal{A}_k)$. Por hipótese de indução, p_k é o $k^{\underline{\text{ésimo}}}$ trajecto mais curto, portanto $p^\star = p_{k+1}$, isto é, $p^\star$ é o $(k+1)^{\underline{\text{ésimo}}}$ trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$. □

É de notar, mais uma vez, que a representação dos nós alternativos, v', ao nível da implementação computacional deste algoritmo é realizada com base numa função $h : X \to \mathcal{N}$, onde $X \subseteq \mathbb{N}$, que indica, para cada $v' \in X$ o respectivo nó correspondente na rede original. Neste caso X representa os vários conjuntos de nós, $\mathcal{N}_k$, que vão sendo construídos.

Como foi referido, esta versão do algoritmo foi sucessivamente melhorada sendo a última versão conhecida por algoritmo MS. Estes melhoramentos visam essencialmente tentar reduzir a quantidade de informação a armazenar, o que se traduz também num aumento de eficiência no que respeita ao tempo de execução.

O primeiro melhoramento introduzido permite a redução do número de alternativas de nós que é necessário criar, em cada transformação de uma rede.

Como se pode observar na figura 22, o nó $1'$ não pertence a nenhum trajecto de S para T, dado que não existe em $(\mathcal{N}_2, \mathcal{A}_2) = (\mathcal{N}', \mathcal{A}')$ nenhum trajecto de S para $1'$, pelo que aquele nó nunca será utilizado. O mesmo acontece com o nó $2'$, alternativa de 2 em $(\mathcal{N}_2, \mathcal{A}_2)$. Assim, a rede transformada, $(\mathcal{N}', \mathcal{A}')$, obtida a partir de um dado trajecto p de s para t numa rede $(\mathcal{N}, \mathcal{A})$ pode conter apenas cópias dos nós de p que se seguem ao primeiro nó ao qual chega mais do que um arco. Como é demonstrado no lema 5.14 este é o primeiro nó de p para o qual existe mais do que um trajecto a partir de s em $(\mathcal{N}, \mathcal{A})$, o que implica que para os nós de p anteriores a este é suficiente manter o subtrajecto de p com início em s na rede original, não sendo necessário criar as suas alternativas. Neste lema é demonstrado ainda que este resultado é válido apenas para o primeiro nó, ou seja, que não é possível evitar cópias de nós seguintes ao primeiro ao qual chega mais do que um arco.

Lema 5.14 *Seja $p = \langle v_1, \ldots, v_{\ell_p} \rangle$ um trajecto de s para t numa rede $(\mathcal{N}, \mathcal{A})$ e v_i, com $1 < i \leq \ell_p$, o primeiro nó de p ao qual chega mais do que um arco. Então:*

1. *O nó v_i é o primeiro nó de p para o qual existe mais do que um trajecto com início em s.*

2. *Para os nós que se seguem a v_i em p existe mais do que um trajecto com início em s.*

Demonstração:

1. Comecemos por mostrar que se a um nó de p chega mais do que um arco, então existe mais do que um trajecto de s para esse nó. Para isso consideremos um nó v_i, com $1 < i \leq \ell_p$ do trajecto p, tal que $(j, v_i) \neq (v_{i-1}, v_i)$ é um arco de $(\mathcal{N}, \mathcal{A})$. Por hipótese, $\mathcal{P}_{sj} \neq \emptyset$, logo existe um trajecto q, em $(\mathcal{N}, \mathcal{A})$ de s para j, e $q \diamond \langle j, v_i \rangle$ é um trajecto de s para v_i. Como $(j, v_i) \neq (v_{i-1}, v_i)$, o trajecto definido é distinto de $\text{sub}_p(s, v_i)$, ou seja, existe mais do que um trajecto de s para v_i.

   Consideremos agora que v_i é o primeiro nó de p naquelas condições, e que v_j, com $1 < j < i$, é outro nó de p anterior a v_i ao qual chega um só arco. Suponhamos que existe um trajecto q, de s para v_j em $(\mathcal{N}, \mathcal{A})$, diferente de $\text{sub}_p(s, v_j)$. Designemos por x o nó de $\text{sub}_p(s, v_j)$ e de q mais distante de v_j, tal que $\text{sub}_q(x, v_j) = \text{sub}_p(x, v_j)$ e anterior a v_j. Como $q \neq \text{sub}_p(s, v_j)$, x é um nó seguinte a s. Então $\text{sub}_p(s, x)$ e $\text{sub}_q(s, x)$ são dois trajectos distintos de s para x e existe mais do que um arco terminando em x, pelo que v_i não seria o primeiro nestas condições.

2. Sendo v_i o primeiro nó de p ao qual chega mais do que um arco, como foi demonstrado em 1, existem trajectos $q_1, \ldots, q_k$ de s para v_i, com $k > 1$. Seja v_j um nó de p que se segue a v_i. Então $\text{sub}_p(v_i, v_j)$ é um trajecto de v_i para v_j e, consequentemente, $q_1 \diamond \text{sub}_p(v_i, v_j), \ldots, q_k \diamond \text{sub}_p(v_i, v_j)$ são k trajectos de s para v_j, ou seja, existem vários trajectos de s para v_j. $\square$

De acordo com este lema, ao construir a rede $(\mathcal{N}_2, \mathcal{A}_2)$ apresentada na figura 22 podemos desprezar, isto é, não copiar, os nós $1'$ e $2'$, uma vez que 3 é o primeiro nó de p_1 no qual termina mais do que um arco.

Seja p'_k o caminho mais curto em $(\mathcal{N}_k, \mathcal{A}_k)$, correspondente ao $k^{\underline{\text{ésimo}}}$ trajecto mais curto em $(\mathcal{N}_1, \mathcal{A}_1)$. De acordo com este lema, basta acrescentar a $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$ os nós de p'_k a partir do primeiro ao qual chega mais do que um arco. Este primeiro nó terá um papel semelhante ao do primeiro nó copiado antes do melhoramento do algoritmo. Assim, designando este nó por v_i, de modo a que p'_k não se possa realizar em $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$, em v'_i devem terminar os arcos que terminavam em v_i, à excepção daquele utilizado em p'_k.

Segundo o algoritmo original de Martins, após a obtenção da rede transformada é utilizado um algoritmo para determinar o caminho mais curto nesta rede. Aplicando este raciocínio à rede $(\mathcal{N}_2, \mathcal{A}_2)$ da figura 22 obtemos, um rótulo consistindo num par (π_i^s, ξ_i^s), associado a cada nó i, que apresentamos na tabela 2.

$i \in \mathcal{N}_2$	S	1	2	3	4	5	T	$3'$	$5'$
(π_i^s, ξ_i^s)	(0,–)	(0,S)	(0,1)	(0,2)	(0,3)	(3,3)	(4,$5'$)	(1,1)	(4,$3'$)

Tabela 2

É de notar que os rótulos dos nós da rede original, à excepção do rótulo de T, não foram alterados. Assim, os novos nós podem ser rotulados correctamente aquando da sua criação, analisando os arcos que terminam em cada um eles. Deste modo, cada novo nó j é rotulado com o custo

$$\pi_j^s = \min\{\pi_i^s + c_{ij} : i \in \mathcal{R}(j)\},$$

onde $\mathcal{R}(j)$ é o conjunto dos nós da nova rede em que se iniciam arcos que terminam em j. É ainda de notar que, uma vez que o nó terminal t não é copiado, o rótulo deste nó tem de ser alterado. Cada rótulo determinado corresponde ao custo de cada trajecto p_k obtido. Como foi referido, e como é demonstrado em seguida, os rótulos assim calculados são os correctos.

Lema 5.15 *Seja* $\{(\mathcal{N}_1, \mathcal{A}_1),\ldots,(\mathcal{N}_K, \mathcal{A}_K)\}$ *a sequência de redes que se obtém com o algoritmo de Martins, e seja* p'_k *o caminho mais curto de* s *para* t *em* $(\mathcal{N}_k, \mathcal{A}_k)$*, correspondente a* p_k*. Com este algoritmo, os nós que se acrescentam a* $(\mathcal{N}_k, \mathcal{A}_k)$*, para* $k \in \{2, \ldots, K\}$*, ficam rotulados definitivamente.*

Demonstração: Para $k = 1$ os rótulos dos nós de $(\mathcal{N}_1, \mathcal{A}_1)$ são obtidos através da determinação da árvore dos trajectos mais curtos com raiz em s, pelo que são definitivos.

Consideremos agora que os rótulos de $(\mathcal{N}_k, \mathcal{A}_k)$ são definitivos e tentemos mostrar que os rótulos de $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$ também o serão.

Seja $p'_k = \langle v_1, \ldots, v_{\ell'} \rangle$ o caminho mais curto de s para t em $(\mathcal{N}_k, \mathcal{A}_k)$ e seja v_i, com $1 < i \leq \ell'$, o primeiro nó de p'_k ao qual chega mais do que um arco; isto é, v'_i é o primeiro nó a ser acrescentado na construção de $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$. Conforme o raciocínio indicado,

$$\pi^s_{v'_i} = \min\{\pi^s_u + c_{uv'_i} : u \in \mathcal{R}(v'_i)\},$$

onde $\mathcal{R}(v'_i)$ é o conjunto dos nós de $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$ em que se iniciam arcos terminando em v'_i. Pela forma como $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$ é construída, os nós de $\mathcal{R}(v'_i)$ pertencem todos a $(\mathcal{N}_k, \mathcal{A}_k)$. Assim, por hipótese, os nós a partir dos quais v'_i pode ser rotulado, têm rótulos definitivos; portanto o rótulo de v'_i também o é.

Suponhamos agora que v'_j, com $j > i$, tem rótulo definitivo e mostremos que então v'_{j+1} também é rotulado correctamente.

Todos arcos da nova rede terminando em v'_{j+1} têm início em nós da rede $(\mathcal{N}_k, \mathcal{A}_k)$ rotulados correctamente, ou então no nó v'_j, que por hipótese tem também rótulo definitivo, o que conclui a demonstração. □

Resumindo, com base neste melhoramento é construída a rede $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$ e em vez de se calcular o trajecto mais curto em $(\mathcal{N}_{k+1}, \mathcal{A}_{k+1})$, são mantidos os rótulos dos nós originais, enquanto que os novos nós (incluindo o terminal) vão sendo rotulados à medida que são criados, isto é, à medida que são acrescentados à rede.

Consideremos o trajecto $p'_k = \langle v_1, \ldots, v_{\ell'} \rangle$, e seja v_i, com $1 < i < \ell'$, o primeiro nó de p'_k ao qual chega mais do que um arco. Além dos melhoramentos referidos, é ainda possível que não seja necessário duplicar alguns nós de p'_k seguintes a v_i.

De facto, suponhamos que um dado nó de p'_k tem nó alternativo; então, como foi demonstrado no lema 5.15, o rótulo desse nó de p'_k é definitivo pelo que não será alterado. Uma alternativa desse nó seria uma cópia exacta, isto é, com o mesmo rótulo, donde podemos concluir que é desnecessária.

Voltando à rede exemplo, de acordo com os melhoramentos descritos até ao momento, e uma vez que $p_1 = \langle S, 1, 2, 3, 5, T\rangle$, a rede $(\mathcal{N}_2, \mathcal{A}_2)$ da figura 22 não deveria conter os nós $1'$ e $2'$ uma vez que existe um único trajecto de S para aqueles nós. Na figura 23 é representada a rede $(\mathcal{N}_2, \mathcal{A}_2)$ obtida, associando a cada nó i o respectivo rótulo, (π_i^s, ξ_i^s).

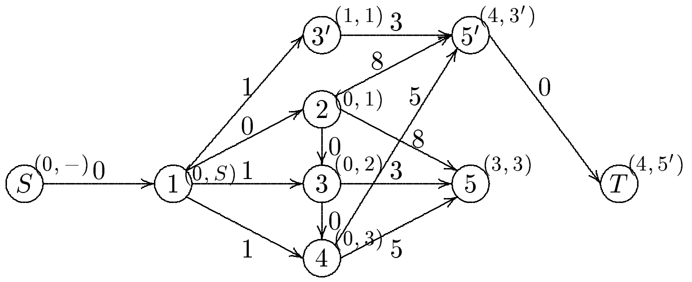

Figura 23

Como podemos observar, $p'_2 = \langle S, 1, 3', 5', T\rangle$ é o caminho mais curto naquela rede. Procedendo de modo análogo é construída a rede $(\mathcal{N}_3, \mathcal{A}_3)$, ficando determinado o trajecto $p'_3 = \langle S, 1, 2, 3, 4, 5'', T\rangle$. Como o nó 3 tem alternativa, isto é, o nó $3'$ pertence a $(\mathcal{N}_3, \mathcal{A}_3)$, então uma nova cópia de 3 teria exactamente as mesmas características que $3'$ pelo que é desnecessária. A rede $(\mathcal{N}_3, \mathcal{A}_3)$, obtida com este novo melhoramento, é apresentada na figura 24. É ainda de notar que na rede original $(\mathcal{N}, \mathcal{A})$, existem apenas dois trajectos de S para 3. O mais curto de entre eles é o representado pelo rótulo do nó 3 em $(\mathcal{N}_3, \mathcal{A}_3)$ e o segundo é o representado pelo rótulo de $3'$. Deste modo não é necessário voltar a copiar aquele nó.

Assim, se um dos nós iniciais de p'_k tiver alternativa o seu rótulo não será alterado, pelo que basta duplicar os nós de p'_k seguintes ou coincidentes com o primeiro que ainda não tem alternativa. Podemos concluir pois que não é necessário copiar os primeiros nós de p'_k nestas condições, reduzindo assim o tamanho da rede construída e, na prática, aumentando a rapidez do algoritmo, como foi referido.

Lema 5.16 *Seja $p'_k = \langle v_1, \ldots, v_{\ell'}\rangle$ o caminho mais curto em $(\mathcal{N}_k, \mathcal{A}_k)$, qualquer que seja $k \in \{2, \ldots, K\}$. Se o nó v_i de p'_k, com $1 < i < \ell'$, tem alternativa, então o mesmo acontece com o nó que o precede, v_{i-1}.*

Demonstração: Seja v_i um nó de p'_k com alternativa. Então v_i foi utilizado em

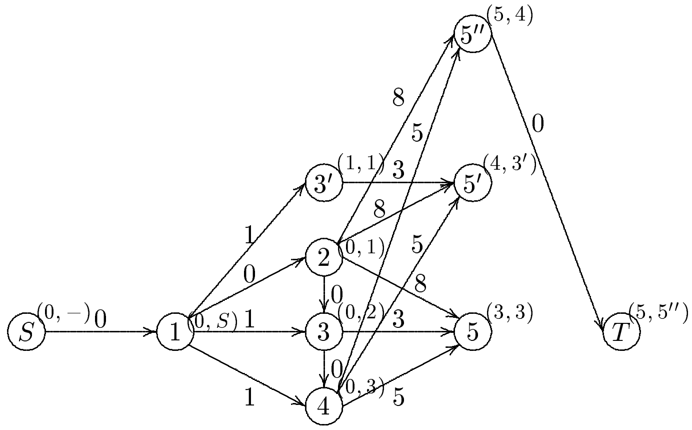

Figura 24

algum caminho mais curto. Como o rótulo de v_i é definitivo, então ele é obtido a partir do mesmo nó. Em p'_k, o nó v_i foi rotulado a partir de v_{i-1} e então este nó também pertenceu a um caminho mais curto, o que significa, como pretendíamos, que tem alternativa. □

De acordo com as alterações realizadas, o algoritmo resultante, da autoria de Azevedo, Madeira, Martins e Pires, [2], pode ser descrito como se mostra no algoritmo 5.7. É de notar que, para cada trajecto p_k, a rotulação do primeiro nó a ser analisado deve ser realizada de modo semelhante à rotulação do nó seguinte ao inicial de p_k, no algoritmo original. Assim, designando aquele nó por v_j^k, o arco (v_{j-1}^k, v_j^k) não deve dar origem a nenhum outro arco na nova rede. Os restantes nós devem ser rotulados como originalmente.

Com base no teorema 5.6 e na forma como é realizada a construção da rede $(\mathcal{N}_k, \mathcal{A}_k)$, para $k \geq 1$, no procedimento 5.7.1 é apresentado *CriarAlternativa(v_i)*, um procedimento para a criação de uma cópia do nó v_i, possivelmente a ser utilizado no algoritmo 5.7.

Observando as figuras que representam as redes obtidas seguindo o algoritmo 5.7, e comparando estas redes com a original, podemos notar que a última destas é ainda constituída por um número de nós e de arcos muito maior do que a original. Podemos verificar ainda que, para cada cópia efectuada de um dado nó, vai diminuindo o número de arcos que nela terminam. Denotemos por $i^{(k)}$ a $k^{\underline{\text{ésima}}}$ cópia de um nó $i \in \mathcal{N}_1$, com $k \geq 1$. Em $i^{(k)}$ terminam os arcos alternativos daqueles que terminavam em $i^{(k-1)}$, excepto, eventualmente, o que antecedia $i^{(k-1)}$ no trajecto determinado. Assim, todos os arcos terminando em $i^{(k)}$ são cópias de arcos que terminam em $i^{(k-1)}$. O último melhoramento realizado, até ao momento, evita as cópias destes

Algoritmo 5.7: *Algoritmo de Azevedo, Madeira, Martins e Pires*

$\mathcal{T}_s \longleftarrow$ árvore dos trajectos mais curtos de s para todos os nós em $(\mathcal{N}, \mathcal{A})$;
$p_1 \longleftarrow \mathcal{T}_s(t)$;
$k \longleftarrow 1$;
`Enquanto` $(k < K)$ e (existem trajectos por determinar) `Fazer`
  $v_i \longleftarrow$ primeiro nó de p'_k ao qual chega mais do que um arco;
  `Se` (i está definido) `Então`
    $v_j \longleftarrow$ primeiro nó de p'_k sem alternativa, tal que $j \geq i$;
    `Para` (todo o $i \in \{j, \ldots, \ell'\}$) `Fazer` *CriarAlternativa($v_i$)*;
    $k \longleftarrow k + 1$;
    $p'_k \longleftarrow$ trajecto mais curto de s para t encontrado;
    $p_k \longleftarrow$ trajecto alternativo de p'_k em $(\mathcal{N}, \mathcal{A})$;
  `Senão`
    não existem mais trajectos por determinar em $(\mathcal{N}, \mathcal{A})$;
  `FimSe`
`FimEnquanto`

arcos, mantendo apenas uma referência, para cada nó cópia, que indica os arcos que lhe estão associados.

De acordo com este melhoramento podemos considerar uma segunda versão do procedimento *CriarAlternativa(v_i)*, que originou o algoritmo de Martins e Santos, [17], reduzindo o espaço de memória ocupado. Esta nova versão é apresentada no procedimento 5.7.2.

Seguindo o algoritmo MS (de Martins e Santos), a rede $(\mathcal{N}_3, \mathcal{A}_3)$ da figura 24 passaria a ter a forma apresentada na figura 25.

É de notar que o rótulo de cada cópia de um nó i indica o custo de um trajecto

Procedimento 5.7.1: *Procedimento CriarAlternativa(v_i) – Versão I*

`Se` (v_i é o primeiro nó a ser copiado) `Então`
  $\mathcal{N}' \longleftarrow \mathcal{N} \cup \{v'_i\}$;
  $\mathcal{F}'(v'_i) \longleftarrow \{(u, v'_i) : (u, v_i) \in \mathcal{F}(v_i) - \{(v_{i-1}, v_i)\}\}$;
`Senão`
  $\mathcal{N}' \longleftarrow \mathcal{N}' \cup \{v'_i\}$;
  $\mathcal{F}'(v'_i) \longleftarrow \{(u, v'_i) : (u, v_i) \in \mathcal{F}(v_i) - \{(v_{i-1}, v_i)\}\} \cup \{(v'_{i-1}, v'_i)\}$;
`FimSe`
$\pi^s_{v'_i} \longleftarrow \min\{\pi^s_u + c_{uv'_i} : (u, v'_i) \in \mathcal{F}'(v'_i)\}$

Procedimento 5.7.2: *Procedimento CriarAlternativa(v_i) – Versão II*

`Se` (v_i é o primeiro nó a ser copiado) `Então`

$\mathcal{N}' \longleftarrow \mathcal{N} \cup \{v_i'\}$;

$\mathcal{F}'(v_i) \longleftarrow \mathcal{F}(v_i) - \{(v_{i-1}, v_i)\}$;

`Senão`

$\mathcal{N}' \longleftarrow \mathcal{N}' \cup \{v_i'\}$;

$\mathcal{F}'(v_i) \longleftarrow (\mathcal{F}'(v_i) - \{(v_{i-1}, v_i)\}) \cup \{(v_{i-1}', v_i)\}$;

`FimSe`

$\pi^s_{v_i'} \longleftarrow \min\{\pi^s_u + c_{uv_i'} : (u, v_i') \in \mathcal{F}'(v_i)\}$

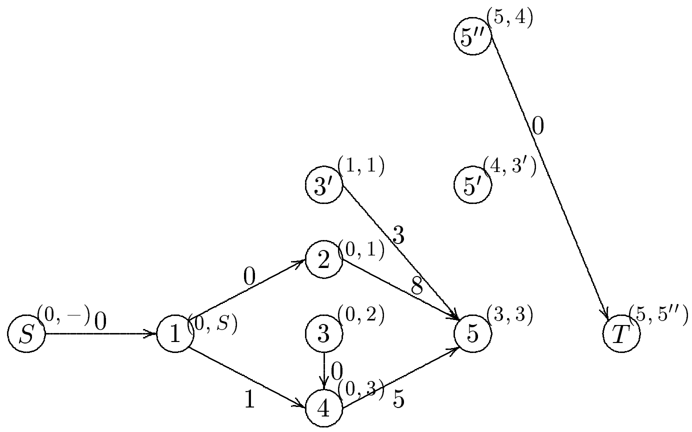

Figura 25

mais curto de s para i e o nó que o antecede nesse trajecto. Tomando como exemplo o nó 5 da rede original, o seu rótulo em $(\mathcal{N}_3, \mathcal{A}_3)$ é $(3, 3)$, correspondente ao trajecto mais curto de S para 5, $\langle S, 1, 2, 3, 5\rangle$, o rótulo de $5'$ corresponde a $\langle S, 1, 3', 5'\rangle$, segundo trajecto mais curto de S para 5, o rótulo de $5''$ ao terceiro trajecto mais curto de S para 5, $\langle S, 1, 2, 3, 4, 5''\rangle$, e assim sucessivamente, aplicando novamente o algoritmo.

Deste modo o número de arcos da rede não aumenta, podendo mesmo diminuir, ao longo do algoritmo.

O pior dos casos para a execução deste algoritmo ocorre, tal como para a generalização do algoritmo de Yen, para uma rede do tipo da figura 19.

Neste caso, para cada p_k é necessário criar novas alternativas para cada nó e determinar o trajecto mais curto na nova rede, o que equivale a rotular essas novas alternativas. Como cada p_k é obtido a partir de p_{k-1} acrescentando novamente todos os arcos da rede, é necessário criar n alternativas e rotulá-las, o que equivale a analisar todos os m arcos da rede. Assim, o algoritmo MS, no pior dos casos tem uma complexidade da $\mathcal{O}(Km)$ no que respeita ao tempo de execução, depois de determinada a árvore dos trajectos mais curtos. No que respeita ao espaço de memória a complexidade é da $\mathcal{O}(Kn)$, no pior dos casos.

6 Enumeração dos K Caminhos Mais Curtos

Nesta secção será apresentado o problema dos K Caminhos Mais Curtos de forma um pouco mais aprofundada, sendo ainda descritos vários algoritmos para a sua resolução.

Como foi referido, neste problema, para um dado K, pretendemos determinar os primeiros K caminhos de s para t, por ordem não decrescente de distâncias. Isto é, pretendemos determinar um conjunto de K caminhos $\{p_1, \ldots, p_K\} \subseteq \mathcal{P}$ tais que $c(p_k) \leq c(p_{k+1})$, sendo p_k determinado exactamente antes de p_{k+1}, para todo o $k \in \{1, \ldots, K-1\}$, e $c(p_K) \leq c(p)$, para todo o $p \in \mathcal{P} - \{p_1, \ldots, p_K\}$.

Uma vez que pretendemos obter apenas caminhos, cada p_k determinado, com $k \in \{1, \ldots, K\}$, não pode conter ciclos, o que torna este problema diferente do problema dos K Trajectos Mais Curtos, dado que alguns dos trajectos determinados na resolução deste último, em geral, não pertencem à solução do primeiro.

Uma consequência deste facto é que, apesar de este ser um subproblema do problema dos K Trajectos Mais Curtos, não verifica o Princípio de Optimalidade; isto é, é possível que o $k^{\underline{\text{ésimo}}}$ caminho óptimo seja constituído por alguns subcaminhos $r^{\underline{\text{ésimos}}}$ óptimos, com $r > k$. É de referir que dado um caminho p, designamos por um **subcaminho** de p um subtrajecto de p, o que significa que é também um caminho, denotando ambos da mesma forma.

Para exemplificar esta situação podemos considerar a rede representada na figura 26.

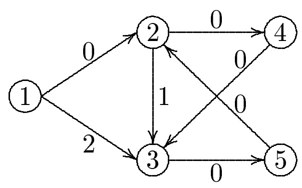

Figura 26

Sendo $s = 1$ e $t = 2$, os caminhos de s para t naquela rede são $p_1 = \langle 1, 2\rangle$, com custo $c(p_1) = 0$, e $p_2 = \langle 1, 3, 5, 2\rangle$, com custo $c(p_2) = 2$.

Por outro lado, nesta rede existem três caminhos de 1 para 3, $t_1 = \langle 1, 2, 4, 3\rangle$,

$t_2 = \langle 1, 2, 3\rangle$ e $t_3 = \langle 1, 3\rangle$, com custos $c(t_1) = 0$, $c(t_2) = 1$ e $c(t_3) = 2$, respectivamente.

Ora, t_3 é o terceiro caminho mais curto de 1 para 3 e é subcaminho de p_2, o segundo caminho mais curto de 1 para 2, o que significa que neste caso o Princípio de Optimalidade não se verifica.

É possível verificar um resultado análogo se a rede for não orientada, podendo ainda a rede anterior, considerando os arcos como não orientados, ser utilizada como contra-exemplo.

Por esta razão este problema foi, durante muito tempo, considerado de resolução mais difícil que o dos K Trajectos Mais Curtos, dado que assim não é possível utilizar o Princípio de Optimalidade como base para um algoritmo capaz de enumerar os K caminhos mais curtos.

Uma possível forma de resolver o presente problema seria utilizar um qualquer algoritmo para determinação dos K trajectos mais curtos, como os descritos na secção 5, seleccionando apenas caminhos de entre os trajectos calculados, até obter os K pretendidos.

Se atendermos ao facto de que, como acontecia com o conjunto dos trajectos entre dois nós, podemos organizar o conjunto dos caminhos entre dois nós sob a forma de uma árvore, então é possível desenvolver algoritmos para a resolução do problema dos K Caminhos Mais Curtos que têm por base a construção dessa estrutura.

Tal como acontecia para os trajectos entre dois nós s e t numa rede $(\mathcal{N}, \mathcal{A})$, podemos considerar uma árvore de caminhos de s para t. De facto, tendo em conta que um caminho é também um trajecto, podemos organizar o conjunto dos caminhos de uma rede tal como foi realizado para o conjunto dos trajectos, formando uma árvore a que chamaremos **árvore dos caminhos de s para t em** $(\mathcal{N}, \mathcal{A})$. Esta árvore é constituída por uma parte da árvore dos trajectos de s para t e, uma vez que o número de caminhos de s para t em $(\mathcal{N}, \mathcal{A})$ é finito, é sempre possível construir a árvore de todos os caminhos numa rede entre dois dos seus nós.

Consideremos novamente a rede apresentada na figura 5, com $s = 1$ e $t = 5$, mas supondo agora que cada arco entre dois nós é não orientado. A esta rede podemos associar a árvore dos caminhos de s para t em $(\mathcal{N}, \mathcal{A})$ que é apresentada na figura 27.

Mais uma vez, cada caminho p_k, com $k \in \{1, \dots, K\}$, é caracterizado, por analogia com a estrutura daquela árvore, por uma parte inicial, um nó desvio e uma parte final. A parte inicial de p_k é o seu subcaminho desde s até ao nó desvio, $v^k_{\alpha_k}$, onde $\alpha_k \in \{1, \dots, \ell_k - 1\}$. A parte final do caminho p_k é o subcaminho de p_k com início em $v^k_{\alpha_k}$ e terminando em t. É de referir que no caso de $\alpha_k = 1$, o nó desvio coincide com o nó inicial s, e então a parte inicial de p_k é constituída apenas por esse nó.

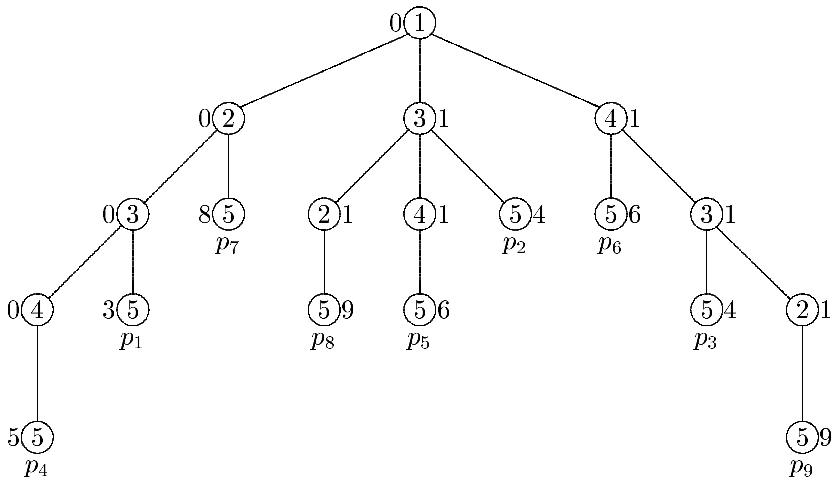

Figura 27

Como referimos anteriormente, neste problema pretendemos obter alguns dos caminhos de s para t, ou seja, pretendemos construir parte da árvore dos caminhos de s para t, constituída apenas pelos K mais curtos. É possível que para isso seja necessário construir uma sobreárvore desta última.

Os algoritmos apresentados nos próximos parágrafos são baseados na construção da árvore dos caminhos de s para t em $(\mathcal{N}, \mathcal{A})$.

O primeiro algoritmo, algoritmo de pesquisa exaustiva, realiza uma construção exaustiva da árvore dos caminhos, tentando obter todos os possíveis trajectos de s para t, desde que não formem ciclos, isto é, de modo a que apenas sejam determinados caminhos.

De seguida são apresentados o algoritmo de Yen e o algoritmo de Katoh, Ibaraki e Mine, que determinam sucessivamente caminhos a partir de alterações convenientes da rede original.

Os algoritmos de Eppstein e MPS utilizados para a enumeração dos K trajectos mais curtos de s para t em $(\mathcal{N}, \mathcal{A})$, podem ser adaptados, ou particularizados, de modo a minimizar a determinação de trajectos com ciclos, e calculando os K caminhos mais curtos. Estas adaptações são também descritas mais adiante.

É de notar que os algoritmos apresentados em seguida são válidos quer para redes orientadas quer para redes não orientadas, exceptuando o algoritmo de Katoh, Ibaraki e Mine, aplicável apenas em redes não orientadas.

6.1 Algoritmo de Pesquisa Exaustiva

Nesta subsecção iremos descrever, muito sucintamente, um algoritmo de pesquisa exaustiva para a determinação ordenada dos K caminhos mais curtos entre dois nós de uma rede.

De um modo geral, o funcionamento deste algoritmo é semelhante ao utilizado para trajectos e apresentado no algoritmo 5.1. Assim, como o nome indica, é realizada a construção exaustiva da árvore dos caminhos, tentando-se obter todos os trajectos possíveis de s para t. No entanto, e uma vez que pretendemos determinar apenas caminhos, a inserção de um novo nó na árvore construída até um certo momento está condicionada pelo facto de este nó ir ou não formar um ciclo com os nós dessa árvore. Se o nó em questão formar um ciclo, então passaríamos a ter na árvore um trajecto que não seria um caminho e assim, dado que pretendemos determinar apenas caminhos, esse nó não é acrescentado.

O algoritmo de pesquisa exaustiva para a determinação dos K caminhos mais curtos de s para t em $(\mathcal{N}, \mathcal{A})$ é descrito no algoritmo 6.1.

Consideremos que, com este algoritmo, pretendíamos inserir na árvore um nó v a partir de outro nó u. Para verificar se v forma um ciclo na árvore dos caminhos construída até esse passo, basta analisar os nós da árvore que pertencem ao ramo onde este pretende ser inserido. Assim, devem ser analisados os nós $i_1, i_2, \ldots, i_\ell$, onde $h(i_1) = u$, $i_2 = \xi^s_{i_1}$, $\ldots$, $i_\ell = \xi^s_{i_{\ell-1}} = 1$ (com $h(1) = s$). Caso $h(i_j) = v$, para algum $j \in \{1, \ldots, \ell\}$, v forma um ciclo e então não é acrescentado à árvore. Este raciocínio é descrito no procedimento 6.1.1, a utilizar no algoritmo 6.1.

Ao contrário do que acontecia com o algoritmo de pesquisa exaustiva para os K trajectos mais curtos, com este algoritmo podemos admitir ciclos na rede, uma vez que são ignorados todos os trajectos que possam surgir e que não são caminhos.

Claro que, sendo este algoritmo baseado numa pesquisa exaustiva, está implícita a construção de toda a árvore dos caminhos de s para t em $(\mathcal{N}, \mathcal{A})$. De facto, os caminhos daquela árvore não são calculados de forma ordenada, o que obriga a que só no final da construção da árvore seja possível seleccionar ordenadamente os K mais curtos, de entre todos os caminhos determinados.

Poderíamos proceder de modo análogo partindo da generalização do algoritmo de Dijkstra apresentado em 5.2, obtendo então um outro algoritmo. Para isso bastaria, como foi realizado na pesquisa exaustiva, impedir a formação de ciclos durante a construção da árvore dos caminhos de s para t.

Dada a semelhança existente com o algoritmo de pesquisa exaustiva, não é acrescentada qualquer outra referência à adaptação da generalização do algoritmo de Dijkstra para a determinação ordenada de caminhos.

Algoritmo 6.1: *Algoritmo de pesquisa exaustiva*

$ordem \longleftarrow 1$;
$\pi^s_{ordem} \longleftarrow 0$;
$X' \longleftarrow \{ordem\}$;
$h(ordem) \longleftarrow s$;
`Enquanto` $(X' \neq \emptyset)$ `Fazer`
  $i \longleftarrow$ um dos nós do conjunto X';
  $X' \longleftarrow X' - \{i\}$;
  $u \longleftarrow h(i)$;
  `Se` $(u \neq t)$ `Então`
    `Para` (todo o $(u, v) \in \mathcal{A}$) `Fazer`
      `Se` (não $FormaCiclo(i, v)$) `Então`
        $ordem \longleftarrow ordem + 1$;
        $X' \longleftarrow X' \cup \{ordem\}$;
        $h(ordem) \longleftarrow v$;
        $\pi^s_{ordem} \longleftarrow \pi^s_i + c_{uv}$;
        $\xi^s_{ordem} \longleftarrow i$;
      `FimSe`
    `FimPara`
  `Senão`
    identificar caminho de 1 para i na árvore
  `FimSe`
`FimEnquanto`
identificar os K caminhos mais curtos de s para t em $(\mathcal{N}, \mathcal{A})$;

6.2 Algoritmo de Yen

No parágrafo 5.2.3 foi descrita uma generalização do algoritmo de Yen para enumerar os K trajectos mais curtos entre dois nós numa rede. Como foi referido na altura, o algoritmo originalmente proposto por Yen, [26, 27], tem como objectivo determinar ordenadamente, não os K trajectos mas, os K caminhos mais curtos. De seguida é apresentada a versão original do algoritmo de Yen, sendo evidenciadas as diferenças entre esta versão e a sua adaptação para determinar trajectos, diferenças essas que permitem o cálculo apenas de caminhos.

Como anteriormente, começamos por determinar o trajecto mais curto em $(\mathcal{N}, \mathcal{A})$ de s para t. Sabemos no entanto, como foi demonstrado no lema 3.2, que existe um caminho com o mesmo custo, pelo que podemos considerar, sem perda de ge-

Procedimento 6.1.1: *Procedimento FormaCiclo(i, v)*

```
Para (todo o j ∈ {i, ξ_i^s, ..., 1}) Fazer
    x ⟵ h(j);
    Se (x = v) Então v forma ciclo
FimPara
```

neralidade, que o trajecto determinado é também p_1, o caminho mais curto de s para t em $(\mathcal{N}, \mathcal{A})$. O caminho p_1 pode então ser calculado utilizando um dos algoritmos apresentados para a resolução do problema do Trajecto Mais Curto, sendo em seguida guardado no conjunto auxiliar que designamos por P. Neste conjunto são guardados os caminhos calculados ao longo do algoritmo e que são candidatos ao $k^{\underline{\text{ésimo}}}$ caminho mais curto, com $k \in \{1, \ldots, K\}$.

Enquanto o conjunto P contiver elementos, e enquanto ainda não tiverem sido retirados K caminhos de P, é executado um passo do algoritmo. No passo genérico k, com $k \in \{1, \ldots, K\}$, é escolhido p_k, $k^{\underline{\text{ésimo}}}$ caminho mais curto e elemento de P com menor custo. A partir de p_k são determinados novos caminhos que são acrescentados ao conjunto P.

O procedimento descrito até agora é semelhante ao utilizado na generalização do algoritmo de Yen. A diferença está nos trajectos e/ou caminhos que são determinados a partir de cada p_k. De facto, no algoritmo GY, dado um trajecto p_k, para cada seu nó v_i^k, seguinte ou coincidente com o nó desvio, $v_{\alpha_k}^k$, é determinado um trajecto p_i^k, da forma $p_i^k = \text{sub}_k(s, v_i^k) \diamond \langle v_i^k, j \rangle \diamond q_j$, onde (v_i^k, j) é um arco da rede não pertencente à árvore dos trajectos construída até ao momento pelo algoritmo e onde $q_j = \mathcal{T}_t(j)$ é o trajecto mais curto de j até t em $(\mathcal{N}, \mathcal{A})$.

À primeira vista poderíamos ser levados a pensar que este raciocínio seria também eficaz quando usado na determinação de caminhos, uma vez que podemos considerar q_j como sendo um caminho e que no início do algoritmo temos apenas p_1, que é um caminho. No entanto, podemos verificar facilmente que a concatenação de um ou mais caminhos pode não ser um caminho. Para exemplificar consideremos $p = \langle 1, 2, 3 \rangle$ e $q = \langle 3, 4, 2, 5 \rangle$ que concatenados formam o trajecto $p \diamond q = \langle 1, 2, 3, 4, 2, 5 \rangle$, contendo o ciclo $\langle 2, 3, 4, 2 \rangle$.

Devemos então evitar que, a partir de um caminho p_k, sejam determinados trajectos com ciclos. Para isso, dado um nó v_i^k de p_k, seguinte ou coincidente com o nó desvio, iremos calcular o caminho q_i^k, que deve ser o mais curto de v_i^k para t em $(\mathcal{N}, \mathcal{A})$, sujeito a certas restrições. De facto, de modo a que $\text{sub}_k(s, v_i^k) \diamond q_i^k$ não contenha ciclos, não devemos permitir que q_i^k contenha nós do caminho $\text{sub}_k(s, v_i^k)$. Além disso, é necessário garantir que a partir do caminho p_k e do nó v_i^k não sejam calculados caminhos repetidos, isto é, caminhos anteriormente determinados.

Com este objectivo podemos adoptar um procedimento análogo ao utilizado para trajectos. De facto, sendo $i \in \{\alpha_k, \dots, \ell_k - 1\}$, e considerando que na rede $(\mathcal{N}, \mathcal{A})$ foram removidos os nós do caminho $\text{sub}_k(s, v_{i-1}^k)$, pretendemos determinar um caminho q_i^k, de v_i^k para t, na rede alterada, que seja o mais curto e tal que o seu primeiro arco não pertença à árvore dos caminhos construída até ao presente passo do algoritmo. Assim, mais uma vez, queremos calcular o caminho de menor custo no conjunto $P^k(i)$, para a rede $(\mathcal{N}, \mathcal{A})$ transformada, definido no parágrafo 5.2.3. Esse caminho será da forma $q_i^k = \langle v_i^k, j \rangle \diamond q_j$, onde o arco (v_i^k, j) não pertence à árvore dos caminhos no passo k do algoritmo, e onde q_j é o caminho mais curto de j para t na rede transformada.

Uma estratégia para a determinação de q_i^k consiste simplesmente em remover da rede os arcos do conjunto $g(\mathcal{A}^k(u))$, isto é, os arcos da árvore construída com início em u, onde u é o nó da árvore que corresponde ao nó v_i^k de p_k, logo $h(u) = v_i^k$, e à rede obtida aplicar um algoritmo para determinação do caminho mais curto entre v_i^k e t.

Outra estratégia possível consiste em resolver o problema do caminho mais curto de todos os nós para t na rede não contendo $s = v_1^k, \dots, v_{i-1}^k$ e procurar o caminho $q_i^k = \langle v_i^k, j \rangle \diamond q_j$ de menor custo, tendo em conta que o arco (v_i^k, j) não deve pertencer a $P^k(i)$.

Na generalização do algoritmo de Yen utilizámos esta última estratégia, uma vez que era possível tirar partido da construção inicial da árvore dos trajectos mais curtos de todos os nós para t em $(\mathcal{N}, \mathcal{A})$. Como vimos, a determinação de caminhos implica alterações da rede, pelo que aquela árvore não pode ser utilizada durante todo o algoritmo quando pretendemos determinar os K caminhos mais curtos. Outra razão para aquela escolha é o facto de a remoção de arcos da rede eliminar a hipótese de serem calculados alguns trajectos na rede. É de notar que neste problema isto não acontece. De facto, ao determinar os caminhos mais curtos, os arcos removidos têm início no nó v_i^k que não deve ser novamente utilizado, uma vez que isso significaria que havia sido calculado um trajecto com pelo menos um ciclo.

O algoritmo de Yen para a enumeração dos K caminhos mais curtos entre dois nós de uma rede que foi aqui descrito é apresentado no algoritmo 6.2.

O ciclo `Enquanto` deste algoritmo é executado K vezes, uma para cada caminho p_k, com $k \in \{1, \dots, K\}$, ou menos se não existirem K caminhos de s para t em $(\mathcal{N}, \mathcal{A})$. Para cada caminho p_k, e para todo o seu nó, v_i^k, seguinte ao nó desvio, é

Algoritmo 6.2: *Algoritmo de Yen*

$p \longleftarrow$ caminho mais curto de s para t em $(\mathcal{N}, \mathcal{A})$;
$P \longleftarrow \{p\}$;
$k \longleftarrow 1$;
`Enquanto` $(k \leq K)$ e $(P \neq \emptyset)$ `Fazer`
  $p_k \longleftarrow$ caminho de P tal que $c(p_k) \leq c(p)$, qualquer que seja $p \in P$;
  $P \longleftarrow P - \{p_k\}$;
  $\alpha_k \longleftarrow$ ordem do nó desvio de p_k;
  `Para` (todo o $i \in \{\alpha_k, \ldots, \ell_k - 1\}$) `Fazer`
    $u \longleftarrow$ nó da árvore tal que $h(u) = v_i^k$;
    $(\mathcal{N}', \mathcal{A}') \longleftarrow$ transformada de $(\mathcal{N}, \mathcal{A})$ removendo os nós $v_1^k, \ldots, v_{i-1}^k$ e arcos de $g(\mathcal{A}^k(u))$;
    $q_i^k \longleftarrow$ trajecto mais curto em $(\mathcal{N}', \mathcal{A}')$;
    $p_i^k \longleftarrow \mathrm{sub}_k(s, v_i^k) \diamond q_i^k$;
    $P \longleftarrow P \cup \{p_i^k\}$;
  `FimPara`
  $k \longleftarrow k + 1$;
`FimEnquanto`

necessário preparar a rede e utilizar um algoritmo para a determinação do trajecto mais curto entre dois nós.

A preparação da rede é, no pior dos casos, da $\mathcal{O}(n)$, enquanto que a determinação do caminho mais curto utilizando o algoritmo de Dijkstra é, no pior dos casos, da $\mathcal{O}(n^2)$, como foi referido na secção 3. Assim sendo, para cada nó de um caminho o número de operações realizadas é da $\mathcal{O}(n^2)$.

No pior dos casos os K caminhos mais curtos são constituídos por todos os nós da rede, ou seja, contêm n nós, e o respectivo nó desvio é o nó inicial, s. Assim, para cada p_k é necessário analisar n nós e, deste modo, no algoritmo de Yen são realizadas $K \times n \times n^2$ operações; ou seja, este algoritmo tem uma complexidade da $\mathcal{O}(Kn^3)$.

6.3 Particularização dos Algoritmos de Eppstein e MPS

Nos parágrafos 5.2.4 e 5.2.5 foram descritos dois algoritmos para a determinação dos K trajectos mais curtos entre dois nós numa rede, o algoritmo de Eppstein e o algoritmo MPS. Estes algoritmos podem ser adaptados de modo a que sejam determinados apenas trajectos sem ciclos, ou seja, caminhos. Nesta subsecção serão apresentadas estas particularizações.

Admitamos que foram determinados os trajectos mais curtos $p_1, \ldots, p_k$ e seja v_i^k um dos nós de p_k, tal que $\alpha_k < i < \ell_k$, onde α_k denota a ordem do nó desvio de p_k e ℓ_k a ordem do seu último nó. No algoritmo original de Eppstein, a partir de v_i^k obtêm-se trajectos da forma

$$p_i^k = \mathrm{sub}_k(s, v_i^k) \diamond \langle v_i^k, j \rangle \diamond q_j,$$

onde $j \in \mathcal{D}(v_i^k) - \{v_{i+1}^k\}$, e q_j é o trajecto mais curto de j para t em $(\mathcal{N}, \mathcal{A})$. Relembremos ainda que numa fase inicial é calculada a árvore, $\mathcal{T}_t$ dos trajectos mais curtos de todos os nós para t em $(\mathcal{N}, \mathcal{A})$ e, para todo o arco (i, j) da rede, o seu custo, c_{ij}, é substituído pelo respectivo custo reduzido $\bar{c}_{ij} = \pi_j^t - \pi_i^t + c_{ij}$, onde π_x^t, com $x \in \mathcal{N}$, denota o custo do melhor caminho de x para t em $(\mathcal{N}, \mathcal{A})$, ou seja, o rótulo de x na árvore determinada. A determinação daquela árvore permite o conhecimento imediato dos caminhos q_j em qualquer passo do algoritmo, enquanto que a mudança de variável simplifica o acesso aos trajectos que vão sendo calculados.

No entanto, como referimos no parágrafo anterior, a concatenação de dois caminhos não é, em geral, um caminho. Por esta razão, caso se pretenda utilizar a árvore com raiz em t, podem ser acrescentados trajectos que não são caminhos à árvore construída ao longo do algoritmo, ou seja, ao conjunto P. Assim sendo, em cada passo da adaptação do algoritmo de Eppstein para a determinação ordenada de caminhos, dado v_i^k, para cada $j \in \mathcal{D}(v_i^k) - \{v_{i+1}^k\}$, é determinado o trajecto

$$p_i^k = \mathrm{sub}_k(s, v_i^k) \diamond \langle v_i^k, j \rangle \diamond q_j.$$

Como pretendemos calcular os caminhos de menor custo possível, q_j deve pertencer à árvore dos trajectos mais curtos com raiz em t, ou seja, $q_j = \mathcal{T}_t(j)$. Além disso, para não obter caminhos repetidos, o arco da rede (v_i^k, j) deve ser tal que $j \neq v_{i+1}^k$. De forma a minimizar a probabilidade de p_i^k ser um trajecto, devemos ainda escolher j no conjunto $\mathcal{D}(v_i^k)$, de modo que não forme um ciclo; isto é, de modo que $j \neq v_r^k$, para todo o $r \in \{1, \ldots, i\}$.

Apesar de $\mathrm{sub}_k(s, v_i^k) \diamond \langle v_i^k, j \rangle$ ser um caminho, pela forma como foi obtido, p_i^k pode conter ciclos. Quer p_i^k seja um caminho ou não, é acrescentado a P, de onde é retirado o elemento de menor custo em cada passo do algoritmo. Assim, em cada um destes passos é necessário verificar se o trajecto retirado é ou não um caminho. Em caso afirmativo o caminho retirado de P será p_{k+1}, o $(k+1)^{\underline{\text{ésimo}}}$ mais curto. No entanto, não podemos desprezar de imediato os trajectos com ciclos que pertencem a P uma vez que, procedendo deste modo, poderemos estar a impedir a determinação de alguns caminhos.

Consideremos que $p' = \langle v'_1, a'_1, v'_2, \ldots, a'_{\ell'-1}, v'_{\ell'} \rangle$ é um trajecto de s para t escolhido em P que não é um caminho, e seja $v'_{\alpha'}$ o seu nó desvio. Consideremos ainda que $v'_{r'}$ é o primeiro nó de p' que forma um ciclo. Então, para cada v'_i, com $\alpha' < i < r'$, são determinados novos trajectos como anteriormente. Estes trajectos são acrescentados ao conjunto P, embora, novamente, possam não ser caminhos. É de notar que no caso de p' ser um caminho, então todos os nós que se seguem ao nó desvio $v'_{\alpha'}$ têm de ser analisados.

Para exemplificar consideremos a rede da figura 28 onde o nó inicial é $s = 1$ e o nó terminal $t = 5$. A árvore dos trajectos mais curtos de todos os nós para $t = 5$ naquela rede é a representada na figura 29.

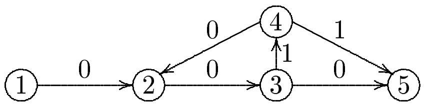

Figura 28

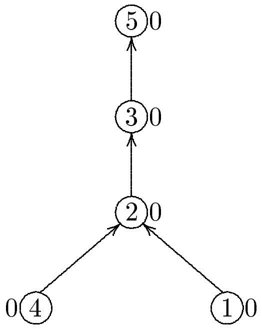

Figura 29

Inicialmente é calculado o caminho mais curto de 1 para 5, pelo que $P = \{p\}$, com $p = \langle 1, 2, 3, 5 \rangle$. O trajecto p é retirado de P e, como não contém ciclos, então $p_1 = p$. A partir dos nós 1, 2 e 3 de p_1 são determinados, se existirem, novos trajectos. Como 3 é o único destes nós a partir do qual sai mais do que um arco, obtemos apenas o trajecto $\langle 1, 2, 3 \rangle \diamond \langle 3, 4 \rangle \diamond \langle 4, 2, 3, 5 \rangle$, onde $\langle 4, 2, 3, 5 \rangle$ é o trajecto mais curto de 4 para 5 – determinado na árvore $\mathcal{T}_t$, calculada inicialmente e que é representada na figura 29. Assim, passaríamos a ter $P = \{q = \langle 1, 2, 3, 4, 2, 3, 5 \rangle\}$. O trajecto q contém o ciclo $\langle 2, 3, 4, 2 \rangle$, pelo que não é um caminho.

Desprezando aquele trajecto, o conjunto P passaria a estar vazio e consequentemente não seriam calculados mais caminhos. No entanto existe um outro caminho de 1 para 5 naquela rede, que é o segundo mais curto, $p_2 = \langle 1, 2, 3, 4, 5 \rangle$. Assim,

Algoritmo 6.3: *Adaptação do algoritmo de Eppstein*

$\mathcal{T}_t \longleftarrow$ árvore dos trajectos mais curtos de todos os nós para t em $(\mathcal{N}, \mathcal{A})$;
`Para` (todo o $(i,j) \in \mathcal{A}$) `Fazer` $\bar{c}_{ij} \longleftarrow \pi_j^t - \pi_i^t + c_{ij}$;
$p \longleftarrow \mathcal{T}_t(s)$;
$P \longleftarrow \{p\}$;
$k \longleftarrow 1$;
`Enquanto` $(k \leq K)$ e $(P \neq \emptyset)$ `Fazer`
  $p' \longleftarrow$ trajecto de P tal que $\bar{c}(p') \leq \bar{c}(p)$, qualquer que seja $p \in P$;
  $P \longleftarrow P - \{p'\}$;
  `Se` $(k = 1)$ `Então` $\alpha' \longleftarrow 0$;
  `Senão` $\alpha' \longleftarrow$ ordem do nó desvio de p';
  $i \longleftarrow \alpha' + 1$;
  `Enquanto` $(i < \ell')$ e (v_i' não forma ciclo com $\mathrm{sub}_{p'}(s, v_{i-1}')$) `Fazer`
    `Para` (todo o $j \in \mathcal{D}(v_i') - \{v_1', \ldots, v_{i+1}'\}$) `Fazer`
      $q_i' \longleftarrow \langle v_i', j\rangle \diamond \mathcal{T}_t(j)$;
      $p_i' \longleftarrow \mathrm{sub}_{p'}(s, v_i') \diamond q_i'$;
      $P \longleftarrow P \cup \{p_i'\}$;
    `FimPara`
    `Se` $(i = \ell' - 1)$ `Então`
      $p_k \longleftarrow p'$;
      $k \longleftarrow k + 1$;
    `FimSe`
    $i \longleftarrow i + 1$;
  `FimEnquanto`
`FimEnquanto`

para cada trajecto com ciclos é necessário analisar os nós seguintes ao nó desvio, mas anteriores ao primeiro nó que se repete e que forma um ciclo. O único nó nestas condições é o nó 4, que quando analisado conduz de imediato ao segundo trajecto mais curto.

A adaptação do algoritmo de Eppstein para enumeração dos K caminhos mais curtos entre os nós s e t em $(\mathcal{N}, \mathcal{A})$ é descrita no algoritmo 6.3. É de relembrar que o conjunto P pode conter trajectos que não são caminhos, o que implica que o ciclo `Enquanto` é, em geral, executado mais do que K vezes.

É ainda de notar novamente que uma vez que a partir de p_1 pretendemos analisar o nó inicial, consideramos que o seu nó desvio tem ordem $\alpha_1 = 0$.

Procedendo de modo análogo é possível particularizar também o algoritmo MPS, por forma a enumerar os K caminhos mais curtos, [18].

Algoritmo 6.4: *Adaptação do algoritmo de Martins, Pascoal e Santos*

$\mathcal{T}_t \longleftarrow$ árvore dos trajectos mais curtos de todos os nós para t em $(\mathcal{N}, \mathcal{A})$;
`Para` (todo o $(i,j) \in \mathcal{A}$) `fazer` $\bar{c}_{ij} \longleftarrow \pi_j^t - \pi_i^t + c_{ij}$;
colocar $(\mathcal{N}, \mathcal{A})$ na *sorted forward star form*;
$p \longleftarrow$ trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$;
$P \longleftarrow \{p\}$;
$k \longleftarrow 1$;
`Enquanto` $(k \leq K)$ e $(P \neq \emptyset)$ `Fazer`
  $p' \longleftarrow$ trajecto de P tal que $\bar{c}(p') \leq \bar{c}(p)$, qualquer que seja $p \in P$;
  $P \longleftarrow P - \{p'\}$;
  $\alpha' \longleftarrow$ ordem do nó desvio de p';
  $i \longleftarrow \alpha'$;
  `Enquanto` $(i < \ell')$ e (v_i' não forma ciclo com $\mathrm{sub}_{p'}(s, v_{i-1}')$) `Fazer`
    $j \longleftarrow$ nó de $\mathcal{D}(v_i') - \{v_1', \ldots, v_i'\}$ tal que $(v_i', j) \in \mathcal{A}$ se segue a a_i';
    `Se` (j está definido) `Então`
      $q_i' \longleftarrow \langle v_i', j \rangle \diamond \mathcal{T}_t(j)$;
      $p_i' \longleftarrow \mathrm{sub}_{p'}(s, v_i') \diamond q_i'$;
      $P \longleftarrow P \cup \{p_i'\}$;
    `FimSe`
    `Se` $(i = \ell' - 1)$ `Então`
      $p_k \longleftarrow p'$;
      $k \longleftarrow k + 1$;
    `FimSe`
    $i \longleftarrow i + 1$;
  `FimEnquanto`
`FimEnquanto`

Resumidamente diremos apenas que para que isso seja possível devem ser seguidos os passos indicados no algoritmo 5.5, aproveitando apenas os trajectos determinados que sejam caminhos. No entanto, mais uma vez, esses trajectos podem ainda originar caminhos, pelo que, como é realizado na adaptação do algoritmo de Eppstein, ao serem retirados do conjunto de candidatos P, devemos acrescentar a P todos os trajectos que poderiam ser obtidos a partir deles. De entre estes trajectos devemos seleccionar apenas aqueles que não contenham o primeiro ciclo, dado que sabemos à partida que os restantes não são caminhos.

A adaptação do algoritmo MPS para enumeração dos K caminhos mais curtos entre s e t numa rede $(\mathcal{N}, \mathcal{A})$ é descrita no algoritmo 6.4.

6.4 Algoritmo de Katoh, Ibaraki e Mine

O algoritmo que iremos descrever nesta subsecção determina ordenadamente os K caminhos mais curtos entre dois nós de uma rede não orientada, na qual $c_{ij} \geq 0$, para todo o arco (i, j). Este algoritmo é devido a Katoh, Ibaraki e Mine, [13], sendo, no que se segue, designado por vezes de forma abreviada por algoritmo de KIM.

Como referimos, $\mathcal{T}_s$ e $\mathcal{T}_t$ representam, respectivamente as árvores dos trajectos mais curtos de s para todos os nós e a árvore de todos os nós para t. É de notar que sendo $(\mathcal{N}, \mathcal{A})$ uma rede não orientada, $\mathcal{T}_t$ coincide com a árvore dos trajectos mais curtos de t para todos os nós, verificando-se o inverso relativamente a $\mathcal{T}_s$. Ainda pelo facto de a rede ser não orientada iremos, propositadamente, identificar um trajecto com o mesmo trajecto mas realizado pela ordem inversa dos nós, dizendo que $\langle v_1, \ldots, v_\ell \rangle = \langle v_\ell, \ldots, v_1 \rangle$. Assim, por exemplo, o trajecto mais curto de s para t é $p_1 = \mathcal{T}_s(t)$ e $p_1 = \mathcal{T}_t(s)$. É de notar que, no caso de existir mais do que um trajecto mais curto na rede, é sempre possível calcular $\mathcal{T}_s$ e $\mathcal{T}_t$ de modo que $p_1 = \mathcal{T}_s(t)$ e $p_1 = \mathcal{T}_t(s)$.

Como referimos, π_i^s e π_i^t denotam, respectivamente, os custos dos trajectos $\mathcal{T}_s(i)$ e $\mathcal{T}_t(i)$, qualquer que seja $i \in \mathcal{N}$.

Consideremos, como exemplo, a rede representada na figura 5, considerando que todos os arcos são não orientados. Para $s = 1$ e $t = 5$, as árvores $\mathcal{T}_s$ e $\mathcal{T}_t$ nesta rede são as representadas na figura 30.

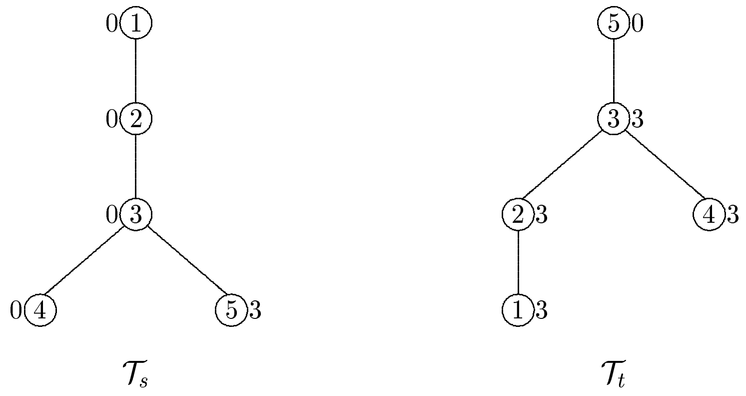

Figura 30

Para estas árvores são apresentados na tabela 3 os valores de π_i^s e π_i^t, para todo o nó $i \in \mathcal{N}$.

Dada a árvore $\mathcal{T}_s$, um seu trajecto $\mathcal{T}_s(i)$ separa-se pela primeira vez de $p_1 = \mathcal{T}_s(t)$ num nó cuja ordem denotamos por β_i^s, qualquer que seja $i \in \mathcal{N}$. Analogamente,

$i \in \mathcal{N}$	1	2	3	4	5
π_i^s	0	0	0	0	3
π_i^t	3	3	3	3	0

Tabela 3

β_i^t representará a ordem do primeiro nó de $\mathcal{T}_t(i)$ que se separa de p_1. Para nós que pertençam ao trajecto p_1, quer β_i^s quer β_i^t são a ordem do nó i em p_1.

Na tabela 4 são apresentados os valores de β_i^s e β_i^t, para todo o $i \in \mathcal{N}$.

$i \in \mathcal{N}$	1	2	3	4	5
β_i^s	1	2	3	3	4
β_i^t	1	2	3	3	4

Tabela 4

O algoritmo de KIM utiliza um procedimento para determinar o caminho mais curto entre dois nós, s e t, de uma rede $(\mathcal{N}, \mathcal{A})$, que coincide em parte com um caminho, $p^\star$, mas que se separa dele antes de um certo nó, $v_\alpha^\star$. A partir deste procedimento podemos, por exemplo, calcular p_2. Este caminho é o mais curto entre s e t diferente de p_1, mas que coincide em parte com p_1 (nem que seja apenas em s). Este procedimento, que denominaremos por $CMC((\mathcal{N},\mathcal{A}),s,t,\text{sub}_{p^\star}(s,v_\alpha^\star))$, é o suporte do algoritmo KIM e será descrito mais adiante. Em seguida apresentamos uma propriedade, na qual se baseia o procedimento *CMC*, e que é demonstrada no lema 6.1.

Lema 6.1 *Se $c_{ij} > 0$ para todo o arco $(i,j) \in \mathcal{A}$, então*

$$\beta_i^s \leq \beta_i^t \tag{1}$$

para todo o nó $i \in \mathcal{N}$. Além disso, se i pertence ao trajecto mais curto de s para t em $(\mathcal{N}, \mathcal{A})$, então $\beta_i^s = \beta_i^t$.

Demonstração: Seja i um nó qualquer do caminho p_1; então, pela definição, $\beta_i^s = \beta_i^t$, uma vez que ambos representam a ordem de i em p_1.

Seja agora i um nó qualquer da rede $(\mathcal{N}, \mathcal{A})$ que não pertence a p_1. Suponhamos que $\beta_i^s > \beta_i^t$, ou seja, que o caminho mais curto de t para i, $\mathcal{T}_t(i)$, se separa de p_1 num nó anterior ao nó no qual $\mathcal{T}_s(i)$ se separa de p_1. Para clarificar, na figura 31 são representados esquematicamente os caminhos p_1, $\mathcal{T}_s(i)$ e $\mathcal{T}_t(i)$.

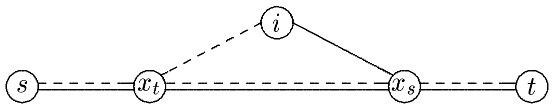

Figura 31

Os traços contínuos desta figura representam caminhos da árvore $\mathcal{T}_s$ e os traços descontínuos caminhos de $\mathcal{T}_t$.

Além do trajecto $\mathcal{T}_s(i) = \mathcal{T}_s(x_s) \diamond \mathcal{T}_s(x_s, i) = \mathcal{T}_s(x_t) \diamond \mathcal{T}_s(x_t, x_s) \diamond \mathcal{T}_s(x_s, i)$, podemos definir em $(\mathcal{N}, \mathcal{A})$ o trajecto $q_s = \mathcal{T}_s(x_t) \diamond \mathcal{T}_t(x_t, i)$, também de s para i. Uma vez que $\mathcal{T}_s(i)$ é o caminho mais curto de s para i, $\pi_i^s \leq c(q_s)$, e então

$$c(\mathcal{T}_s(x_t, x_s)) + c(\mathcal{T}_s(x_s, i)) \leq c(\mathcal{T}_t(x_t, i)). \tag{2}$$

Analogamente, $\mathcal{T}_t(i) = \mathcal{T}_t(i, x_t) \diamond \mathcal{T}_t(x_t) = \mathcal{T}_t(i, x_t) \diamond \mathcal{T}_t(x_t, x_s) \diamond \mathcal{T}_t(x_s)$, é o caminho mais curto de i para t e portanto $\pi_i^t \leq c(q_t)$, onde $q_t = \mathcal{T}_s(i, x_s) \diamond \mathcal{T}_t(x_s)$, logo

$$c(\mathcal{T}_t(x_t, x_s)) + c(\mathcal{T}_t(i, x_t)) \leq c(\mathcal{T}_s(i, x_s)). \tag{3}$$

Além disso, por hipótese, $\beta_{x_s}^s = \beta_{x_s}^t > \beta_{x_t}^t = \beta_{x_t}^s$, e dado que $c_{uv} > 0$, para todo o arco (u, v) da rede, $\pi_{x_t}^s < \pi_{x_s}^s$ e $\pi_{x_s}^t < \pi_{x_t}^t$. Como $\pi_{x_s}^s + c(\mathcal{T}_s(x_s, i)) \leq \pi_{x_s}^s + c(\mathcal{T}_t(x_t, i))$, então

$$c(\mathcal{T}_s(x_s, i)) \leq c(\mathcal{T}_t(x_t, i)).$$

Por outro lado, dado que $c(\mathcal{T}_t(i, x_t)) + \pi_{x_t}^t \leq c(\mathcal{T}_s(i, x_s)) + \pi_{x_t}^t$, analogamente

$$c(\mathcal{T}_t(i, x_t)) \leq c(\mathcal{T}_s(i, x_s)).$$

No entanto, por hipótese a rede $(\mathcal{N}, \mathcal{A})$ é não orientada, portanto $\mathcal{T}_s(x_s, i) = \mathcal{T}_s(i, x_s)$ e $\mathcal{T}_t(i, x_t) = \mathcal{T}_t(x_t, i)$, donde podemos concluir que $c(\mathcal{T}_s(x_s, i)) = c(\mathcal{T}_t(i, x_t))$.

Assim, quer de (2) quer de (3), $c(\mathcal{T}_s(x_t, x_s)) \leq 0$ o que contraria a hipótese. □

Admitindo a existência de arcos na rede com custo nulo podemos ainda, depois de calcular $\mathcal{T}_s$, transformar esta árvore, caso seja necessário, por forma a que a propriedade (1) do lema 6.1 se verifique para todo o nó da rede. Para isso basta rotular cada nó $i \in \mathcal{N}$ de modo a que β_i^s seja mínimo. Por esta razão vamos admitir, sem perda de generalidade, que $c_{ij} \geq 0$, qualquer que seja $(i, j) \in \mathcal{A}$.

Sendo $p^\star$ o caminho mais curto de s para t numa rede $(\mathcal{N}, \mathcal{A})$, com base na propriedade (1) do lema 6.1 é possível dividir em duas classes os caminhos de s para t numa rede $(\mathcal{N}, \mathcal{A})$, que coincidem com $p^\star$ até um dado nó, como é demonstrado no lema 6.2.

Lema 6.2 *Seja $p^\star = \langle v_1^\star, \ldots, v_{\ell_\star}^\star \rangle$ o caminho mais curto entre s e t em $(\mathcal{N}, \mathcal{A})$. Se existirem caminhos de s para t em $(\mathcal{N}, \mathcal{A})$ que se separem de $p^\star$ antes do nó $v_\alpha^\star$, com $\alpha \in \{2, \ldots, \ell_\star\}$, então p, o mais curto destes caminhos, é de uma das seguintes formas:*

1. *$p = \mathcal{T}_s(u) \diamond \mathcal{T}_t(u)$, com $u \in \mathcal{N}$ tal que $\beta_u^s < \alpha$;*

2. *$p = \mathcal{T}_s(u) \diamond \langle u, v \rangle \diamond \mathcal{T}_t(v)$, onde $(u, v) \in \mathcal{A}$ não pertence a $\mathcal{T}_s$ nem a $\mathcal{T}_t$ e com $\beta_u^s < \alpha$.*

Demonstração: Seja $v_\alpha^\star$ um nó de $p^\star$ diferente de s, e suponhamos que existe $p = \langle v_1, \ldots, v_\ell \rangle$, caminho mais curto entre s e t em $(\mathcal{N}, \mathcal{A})$, não contendo $\mathrm{sub}_{p^\star}(s, v_\alpha^\star)$. Então o caminho p separa-se de $p^\star$ num certo nó v_j, ou seja, $\mathrm{sub}_p(s, v_j) = \mathrm{sub}_{p^\star}(s, v_j^\star)$ e $\mathrm{sub}_p(s, v_{j+1}) \neq \mathrm{sub}_{p^\star}(s, v_{j+1}^\star)$. Como p não contém $\mathrm{sub}_{p^\star}(s, v_\alpha^\star)$, então $v_j = v_j^\star$ é anterior a $v_\alpha^\star$ em $p^\star$, logo $j < \alpha$.

Consideremos agora que $(u, v) = (v_k, v_{k+1})$ é o primeiro arco do caminho p que não pertence a $\mathcal{T}_s$. Neste caso $\mathcal{T}_s(u)$ separa-se de $p^\star$ em v_j e assim $\beta_u^s = k < \alpha$. Então é possível escrever p na forma $p = \mathcal{T}_s(u) \diamond \langle u, v \rangle \diamond q$, onde $q = \langle v_{j+1}, \ldots, v_\ell \rangle$.

Se o caminho $\langle u, v \rangle \diamond q$ de u para t estiver na árvore $\mathcal{T}_t$, então coincide com $\mathcal{T}_t(u)$ e p é da forma 1, isto é, $p = \mathcal{T}_s(u) \diamond \mathcal{T}_t(u)$, com u tal que $\beta_u^s < \alpha$. Analogamente, se q estiver em $\mathcal{T}_t$, então $p = \mathcal{T}_s(u) \diamond \langle u, v \rangle \diamond \mathcal{T}_t(v)$, onde (u, v) não pertence a $\mathcal{T}_s$ nem a $\mathcal{T}_t$, e $\beta_u^s < \alpha$, o que significa que p é da forma 2.

Consideremos agora que q não está em $\mathcal{T}_t$. Então $\mathcal{T}_t(v)$ é o caminho mais curto de v para t; logo, sendo $q' = \mathcal{T}_t(v)$, concluímos que $c(q') \leq c(q)$ e $p' = \mathcal{T}_s(u) \diamond \langle u, v \rangle \diamond q'$ é um trajecto de s para t, mais curto do que p e que não contém $\mathrm{sub}_{p^\star}(s, v_\alpha^\star)$.

Suponhamos que p' não é um caminho, ou seja, pelo menos um dos nós de $\mathcal{T}_t(v)$ pertence a $\mathcal{T}_s(u)$. Então, podemos considerar dois casos:

1. $j < \beta_v^t$

   (a) $\mathcal{T}_s(v_j^\star, u)$ e $\mathcal{T}_t(v, v_{\beta_v^t}^\star)$ não têm nós comuns.
   Consideremos $p' = \mathcal{T}_s(u) \diamond \langle u, v \rangle \diamond \mathcal{T}_t(v)$, que é do tipo 1 ou 2 e é caminho. Como $c(\mathcal{T}_t(v)) \leq c(\mathrm{sub}_p(v, t))$, então $c(p') \leq c(p)$.

   (b) $\mathcal{T}_s(v_j^\star, u)$ e $\mathcal{T}_t(v, v_{\beta_v^t}^\star)$ têm nós em comum.
   De entre os nós comuns, seja w o mais próximo de $v_j^\star$ no trajecto $\mathcal{T}_s(v_j^\star, u)$ e consideremos $p' = \mathcal{T}_s(w) \diamond \mathcal{T}_t(w)$, que é caminho e é do tipo 1. Como $c(\mathcal{T}_t(w)) \leq c(\mathrm{sub}_p(w, t)) \leq c(\mathrm{sub}_p(v, t))$, então $c(p') \leq c(p)$.

2. $j \geq \beta_v^t$

   Seja w o nó de $\mathcal{T}_s(v^\star_{\beta_v^s}, v)$ mais próximo de $v^\star_{\beta_v^s}$, de entre aqueles que pertencem simultaneamente a $\mathcal{T}_s(v^\star_{\beta_v^s}, v)$ e a $\text{sub}_p(v, t)$.

   Consideremos o caminho $p' = \mathcal{T}_s(w) \diamond \text{sub}_p(w, t)$. O nó w não pertence a $p^\star$, isto é, $p' \neq p^\star$, uma vez que $\text{sub}_p(v, t)$ não tem nós comuns com o caminho $\mathcal{T}_s(v^\star_{\beta_v^s}) = \text{sub}_{p^\star}(s, v^\star_{\beta_v^s})$. Por outro lado, p' não contém $\text{sub}_{p^\star}(s, v^\star_\alpha)$. Se p' for do tipo 1 ou 2 a demonstração está concluída. Caso contrário podemos aplicar novamente o mesmo argumento, depois de substituir p por p'. Uma vez que o número de arcos de $\text{sub}_p(w, t)$ é finito este processo será também finito. □

Com base neste lema podemos determinar um caminho nas condições enunciadas, analisando todos os possíveis caminhos dos tipos 1 e 2, e escolhendo aquele que tiver menor custo. É de notar que p_2, o segundo caminho mais curto, é o caminho mais curto de s para t em $(\mathcal{N}, \mathcal{A})$ que não coincide com $p_1 = \text{sub}_1(s, t)$. Deste modo, p_2 pode ser determinado a partir da aplicação do lema 6.2.

Consideremos que as árvores $\mathcal{T}_s$ e $\mathcal{T}_t$ estão construídas, e que são conhecidos os valores β_u^s e β_u^t, para qualquer nó $u \in \mathcal{N}$. Para encontrar caminhos do tipo 1 devem ser analisados todos os nós u da rede tais que $\beta_u^s < \alpha$. De modo análogo, para encontrar caminhos do tipo 2 devem ser analisados todos os arcos (u, v), tais que $\beta_u^s < \alpha$. Assim, a determinação de um caminho da forma indicada exige um número de operações da $\mathcal{O}(n + m)$.

A determinação de um caminho entre dois dados nós, s e t, numa rede $(\mathcal{N}, \mathcal{A})$, que se separa de um caminho $p^\star$ antes do respectivo nó de ordem α pode então ser descrita como é indicado no procedimento 6.5.4, apresentado mais adiante.

O modo geral de funcionamento do algoritmo de KIM apresenta algumas semelhanças com o algoritmo de Yen. De facto, inicialmente são calculados p_1 e p_2, os dois caminhos mais curtos de s para t na rede em questão. É utilizado um conjunto P, cujos elementos são os caminhos sucessivamente calculados e candidatos ao $k^{\underline{\text{ésimo}}}$ caminho mais curto, com $k \in \{1, \ldots, K\}$. Em cada passo do algoritmo é retirado de P o seu elemento com menor custo. Consideramos que no passo k, o caminho retirado é p_k, com $k \in \{2, \ldots, K\}$. A partir de p_k, e realizando algumas alterações na rede $(\mathcal{N}, \mathcal{A})$, são determinados outros caminhos, candidatos ao $(k+1)^{\underline{\text{ésimo}}}$ mais curto, a acrescentar ao conjunto P.

Como referimos, o primeiro caminho determinado é p_1, podendo para tal ser utilizado um dos algoritmos apresentados na subsecção 3.1. É de notar que, por conveniência, iremos considerar mais uma vez $\alpha_1 = 0$. Em seguida, aplicando o procedimento *CMC*$((\mathcal{N}, \mathcal{A}), s, t, p_1)$, é calculado o segundo caminho mais curto de s

para t em $(\mathcal{N},\mathcal{A})$, que se separa de p_1 num qualquer dos seus nós. Assim, o caminho resultante, p, é tal que $p = p_2$ e o conjunto P passa a ser $P = \{p\}$.

Na primeira execução do ciclo `Enquanto`, que identificamos com o segundo passo do algoritmo, é retirado o único elemento do conjunto P, p_2, e a partir deste são calculados, se existirem, três novos caminhos, P_a, P_b e P_c. Entre estes deverá estar o terceiro mais curto, p_3.

Como foi referido p_1 e p_2, os dois caminhos determinados até ao momento, contêm uma parte inicial comum, desde o nó inicial, s, até ao nó desvio de p_2, $v^2_{\alpha_2} = v^1_{\alpha_2}$. Pretendemos encontrar neste passo alguns caminhos, que não p_1 e p_2, com o menor custo possível. Estes caminhos devem desviar-se de p_1 e de p_2 em algum dos seus nós. Assim, vamos considerar três hipóteses, o que equivale a dividir os elementos de $\mathcal{P} - \{p_1, p_2\}$ em três grupos, considerando que p_3 se separa de p_1 e p_2 antes ou depois de $v^2_{\alpha_2}$. No segundo caso o primeiro arco de p_3 seguinte a $v^2_{\alpha_2}$ pode coincidir com o de p_1 ou com o de p_2. Pode ainda acontecer que p_3 se separe de p_1 e p_2 precisamente no nó desvio, $v^2_{\alpha_2}$. Agrupando todos estes casos surgem as três hipóteses mencionadas.

Designemos por P_a o caminho mais curto coincidente com p_2 desde s até $v^2_{\alpha_2+1}$, nó seguinte ao nó desvio de p_2, separando-se depois; por P_b o caminho mais curto coincidente com p_1 (e com p_2) de s até $v^2_{\alpha_2}$, diferente de p_2 e separando-se de p_1 num dos nós seguintes a $v^2_{\alpha_2}$; e por fim, por P_c o caminho mais curto que se separa de p_1 (e consequentemente de p_2) antes de $v^2_{\alpha_2}$.

Assim, $P_a = \mathrm{sub}_2(s, v^2_{\alpha_2+1}) \diamond p$ e, como $P_a \neq p_2$, então $p \neq \mathrm{sub}_2(v^2_{\alpha_2+1}, t)$. Além disso p deve ser o caminho mais curto de $v^2_{\alpha_2+1}$ até t, diferente de $\mathrm{sub}_2(v^2_{\alpha_2+1}, t)$, pelo que pode ser calculado pela utilização de $\mathit{CMC}((\mathcal{N},\mathcal{A}), v^2_{\alpha_2+1}, t, \mathrm{sub}_2(v^2_{\alpha_2+1}, t))$. Como referimos anteriormente, a concatenação de dois caminhos nem sempre é um caminho. Por esta razão não devemos permitir que p contenha nós de $\mathrm{sub}_2(s, v^2_{\alpha_2})$. Deste modo os nós de $\mathrm{sub}_2(s, v^2_{\alpha_2})$ e os arcos iniciados ou terminado naqueles nós devem ser retirados de $(\mathcal{N},\mathcal{A})$.

Analogamente, $P_b = \mathrm{sub}_1(s, v^1_{\alpha_2}) \diamond p$, com $p \neq \mathrm{sub}_1(v^1_{\alpha_2}, t)$. Então p pode ser determinado por $\mathit{CMC}((\mathcal{N},\mathcal{A}), v^1_{\alpha_2}, t, \mathrm{sub}_1(v^1_{\alpha_2}, t))$. Mais uma vez, de forma a garantir que P_b seja um caminho, devem ser removidos os nós de $\mathrm{sub}_1(s, v^1_{\alpha_2-1})$. Além disso, P_b não deve coincidir com p_2, pelo que deve ainda ser retirado de $(\mathcal{N},\mathcal{A})$ o arco $(v^2_{\alpha_2}, v^1_{\alpha_2+1})$. É de notar que P_b é o caminho mais curto que coincide com p_1 de s até $v^2_{\alpha_2+1}$, separando-se num dos nós seguintes, ou que coincide com p_1 e p_2 de s até ao nó desvio, separando-se nesse mesmo nó.

Por fim, P_c é tal que a sua parte inicial é diferente da parte inicial de p_1 e p_2, $\mathrm{sub}_1(s, v^1_{\alpha_2}) = \mathrm{sub}_2(s, v^2_{\alpha_2})$. Então P_c resulta da aplicação do procedimento $\mathit{CMC}((\mathcal{N},\mathcal{A}), s, t, \mathrm{sub}_1(s, v^1_{\alpha_2}))$.

Estes três caminhos são armazenados no conjunto P. Aquele que tiver o menor custo é retirado no passo seguinte do algoritmo, correspondendo ao terceiro caminho mais curto. É de notar que quer P_a como P_b ou P_c podem não existir e que, se $\mathrm{sub}_1(s, v^2_{\alpha_2})$ contiver apenas um ou mesmo nenhum arco não faz sentido tentar determinar P_c (neste caso, se $\alpha_2 = 1$ ou $\alpha_2 = 2$).

Consideramos que P_a foi determinado a partir de p_2, enquanto que P_b e P_c foram obtidos a partir de p_1, uma vez que estes são os caminhos mais curtos contendo uma parte inicial coincidente com P_a, P_b ou P_c com maior número de nós.

A forma como é tratado o caminho p_k, com $k \in \{4, \ldots, K\}$, é análoga à utilizada para p_2 e é descrita em seguida.

Genericamente iremos determinar, quando muito, três novos caminhos sempre que é retirado p_k, $k^{\underline{\text{ésimo}}}$ caminho mais curto, para cada $k \in \{3, \ldots, K\}$.

Consideremos que p_k, com $k \in \{3, \ldots, K\}$, foi calculado a partir de p_j, para algum $j \in \{1 \ldots, k-1\}$, ou seja, p_j é o caminho mais curto para o qual se verifica $\mathrm{sub}_k(s, v^k_{\alpha_k}) = \mathrm{sub}_j(s, v^j_{\alpha_k})$ e $v^k_{\alpha_k+1} \neq v^j_{\alpha_k+1}$.

Designemos por W_r, com $1 \leq r \leq K$, o conjunto das ordens dos nós desvio de caminhos p_k, $r < k \leq K$, determinados a partir de p_r.

Seja $\gamma = \min\{\alpha \in W_j : \alpha_k < \alpha \leq \ell_j\}$ a menor ordem de um nó desvio de um caminho calculado a partir de p_j, superior a α_k; isto é, a ordem do nó desvio mais próximo e seguinte a $v^k_{\alpha_k}$, dos caminhos calculados a partir de p_j.

Seja ainda $\delta = \max\{\alpha \in W_j : \alpha_j < \alpha < \alpha_k\}$ a maior ordem de um nó desvio de um caminho calculado a partir de p_j, inferior a α_k; isto é, a ordem do nó desvio mais próximo e anterior a $v^k_{\alpha_k}$, dos caminhos calculados a partir de p_j.

A partir de p_k são calculados, quando muito, três novos caminhos:

- P_a, o caminho mais curto coincidente com p_k de s até $v^k_{\alpha_k+1}$, e que se desvia depois;
- P_b, o caminho mais curto coincidente com p_j de s até $v^k_{\alpha_k}$, e que se desvia depois deste nó mas antes de v^k_γ;
- P_c, o caminho mais curto coincidente com p_k e p_j de s até $v^k_\delta = v^j_\delta$, e que se desvia antes de $v^k_{\alpha_k}$.

Assim, estes três caminhos são, $P_a = \mathrm{sub}_k(s, v^k_{\alpha_k+1}) \diamond p$, onde p é determinado por $CMC((\mathcal{N}', \mathcal{A}'), v^k_{\alpha_k+1}, t, \mathrm{sub}_k(v^k_{\alpha_k+1}, t))$; $P_b = \mathrm{sub}_j(s, v^j_{\alpha_k}) \diamond p$, onde p é o caminho obtido a partir de $CMC((\mathcal{N}', \mathcal{A}'), v^j_{\alpha_k}, t, \mathrm{sub}_j(v^j_{\alpha_k}, v^j_\gamma))$; e, por último, $P_c = \mathrm{sub}_j(s, v^j_\delta) \diamond p$, tal que p é resultante da aplicação do procedimento $CMC((\mathcal{N}', \mathcal{A}'), v^j_\delta, t, \mathrm{sub}_k(v^j_\delta, v^j_{\alpha_k}))$.

Por forma a que sejam determinados apenas caminhos, devem ser realizadas algumas alterações na rede $(\mathcal{N}, \mathcal{A})$. Assim, para calcular:

- P_a devem ser removidos da rede os nós de p_k anteriores e coincidentes com $v^k_{\alpha_k}$, isto é, todos nós do caminho $\text{sub}_k(s, v^k_{\alpha_k}) = \text{sub}_j(s, v^j_{\alpha_k})$;

- P_b devem ser apagados os nós de p_j, que são também nós de p_k, anteriores a $v^j_{\alpha_k}$, ou seja, os nós do caminho $\text{sub}_j(s, v^j_{\alpha_k - 1}) = \text{sub}_k(s, v^k_{\alpha_k - 1})$;

- P_c devem ser apagados de $(\mathcal{N}, \mathcal{A})$ os nós de p_j e de p_k, anteriores a v^j_δ, isto é, todos os nós do caminho $\text{sub}_j(s, v^j_{\delta-1}) = \text{sub}_k(s, v^k_{\delta-1})$.

Por outro lado, de modo a que não sejam calculados caminhos repetidos, devem ainda ser realizadas outras transformações na rede $(\mathcal{N}, \mathcal{A})$, aquando da determinação de P_b e P_c. Desta forma, ao calcular:

- P_b devem ser apagados da rede os arcos da árvore construída pelo algoritmo até ao momento, com início no nó $v^j_{\alpha_k}$;

- P_c devem ser apagados da rede os arcos da mesma árvore, com início no nó v^j_δ.

De acordo com estas modificações são obtidas as redes $(\mathcal{N}', \mathcal{A}')$, utilizadas para na determinação de P_a, P_b e P_c.

Mais uma vez os caminhos calculados são armazenados no conjunto P, sendo candidatos a p_r, com $r > k$. No caso de existirem, consideramos que P_a foi determinado a partir de p_k, e que P_b e P_c foram obtidos a partir de p_j. É ainda de notar que sempre que um novo caminho é calculado, a ordem do respectivo nó desvio deve ser acrescentada a W_k, de modo a que este conjunto seja construído ao longo do algoritmo.

O algoritmo de KIM é descrito resumidamente no algoritmo 6.5.

Em termos do número de operações realizadas, o procedimento 6.5.4 começa por determinar as árvores $\mathcal{T}_s$ e $\mathcal{T}_t$, podendo para tal ser utilizado o algoritmo de Dijkstra apresentado na subsecção 3.1, que como foi referido tem, no pior dos casos, uma complexidade da $\mathcal{O}(n^2)$. Em seguida são calculados os valores β^s_u, β^t_u e ξ^s_u, para todo o $u \in \mathcal{N}$, o que exige um número de operações da $\mathcal{O}(n)$. Além disso, como foi referido anteriormente, a pesquisa do caminho mais curto do tipo 1 ou do tipo 2 é de $\mathcal{O}(n + m)$. Assim, este procedimento tem uma complexidade da $\mathcal{O}(n^2 + m)$.

Relativamente ao algoritmo de KIM propriamente dito, para cada um dos K trajectos mais curtos (supondo que existem) é utilizado, no máximo, três vezes o procedimento *CMC*, o que implica que, desprezando as operações utilizadas para modificar a rede, aquele algoritmo tenha uma complexidade de $\mathcal{O}(Kn^2 + Km)$.

Para clarificar o modo de funcionamento deste algoritmo vamos em seguida apresentar um pequeno exemplo, onde são "simulados" alguns dos seus passos, aquando

Algoritmo 6.5: *Algoritmo de Katoh, Ibaraki e Mine*

$p_1 \longleftarrow$ caminho mais curto de s para t em $(\mathcal{N}, \mathcal{A})$;
$\alpha_1 \longleftarrow 0$;
$W_1 \longleftarrow \{1, \ell_1\}$;
$p \longleftarrow CMC((\mathcal{N}, \mathcal{A}), s, t, p_1)$;
$P \longleftarrow \{p\}$;
$k \longleftarrow 2$;
`Enquanto` $(k \leq K)$ e $(P \neq \emptyset)$ `Fazer`
  $p_k \longleftarrow$ caminho de P tal que $c(p_k) \leq c(p)$, qualquer que seja $p \in P$;
  $P \longleftarrow P - \{p_k\}$;
  $j \longleftarrow$ valor de $\{1, \ldots, k-1\}$ tal que p_k foi calculado a partir de p_j;
  $\alpha_k \longleftarrow$ ordem do nó desvio de p_k;
  $W_k \longleftarrow \{\alpha_k + 1, \ell_k\}$;
  $CalcularPa((\mathcal{N}, \mathcal{A}), p_k)$;
  $P \longleftarrow P \cup \{P_a\}$;
  $\gamma \longleftarrow \min\{\alpha \in W_j : \alpha_k < \alpha \leq \ell_j\}$;
  $CalcularPb((\mathcal{N}, \mathcal{A}), p_k)$;
  $P \longleftarrow P \cup \{P_b\}$;
  `Se` $(\alpha_k \notin W_j)$ e $(\alpha_k > \alpha_j + 1)$ `Então`
    $W_j \longleftarrow W_j \cup \{\alpha_k\}$;
    $\delta \longleftarrow \max\{\alpha \in W_j : \alpha_j < \alpha < \alpha_k\}$;
    $CalcularPc((\mathcal{N}, \mathcal{A}), p_k)$;
    $P \longleftarrow P \cup \{P_c\}$;
  `FimSe`
  $k \longleftarrow k + 1$;
`FimEnquanto`

Procedimento 6.5.1: *Procedimento CalcularPa(*$(\mathcal{N},\mathcal{A})$*,*$p_k$*)*
apagar nós de $\text{sub}_k(s, v^k_{\alpha_k})$;
$p \longleftarrow CMC((\mathcal{N},\mathcal{A}), v^k_{\alpha_k+1}, t, \text{sub}_k(v^k_{\alpha_k+1}, t))$;
$P_a \longleftarrow \text{sub}_k(s, v^k_{\alpha_k+1}) \diamond p$; repor rede original;

Procedimento 6.5.2: *Procedimento CalcularPb(*$(\mathcal{N},\mathcal{A})$*,*$p_k$*)*
apagar nós de $\text{sub}_j(s, v^j_{\alpha_k-1})$;
apagar arcos de $\mathcal{F}(v^j_{\alpha_k})$ de caminhos calculados a partir de p_j, que se desviam em $v^j_{\alpha_k}$;
$p \longleftarrow CMC((\mathcal{N},\mathcal{A}), v^j_{\alpha_k}, t, \text{sub}_j(v^j_{\alpha_k}, v^j_\gamma))$;
$P_b \longleftarrow \text{sub}_j(s, v^j_{\alpha_k}) \diamond p$;
repor rede original;

da sua aplicação à rede representada na figura 5, considerando novamente os seus arcos como sendo não orientados. Lembremos que o nosso objectivo consiste em construir uma parte, ou mesmo a totalidade, da árvore dos caminhos mais curtos de $s = 1$ para $t = 5$ naquela rede, árvore essa representada na figura 27.

No início do algoritmo é calculado o caminho mais curto de 1 para 5 em $(\mathcal{N},\mathcal{A})$, $p_1 = \langle 1, 2, 3, 5\rangle$ e em seguida é determinado $p = \langle 1, 3, 5\rangle$, segundo mais curto, utilizando o procedimento *CMC*$((\mathcal{N},\mathcal{A}), s, t, p_1)$. Neste momento o conjunto P é constituído apenas pelo caminho p.

Na primeira execução do ciclo `Enquanto` é retirado o caminho $p_2 = \langle 1, 3, 5\rangle$ de P, que tem por nó desvio o nó inicial, $s = 1$. Para a determinação de P_a é necessário realizar novamente uma chamada do procedimento *CMC*$((\mathcal{N}', \mathcal{A}'), 3, 5, \langle 3, 5\rangle)$, que irá calcular $p = \langle 3, 4, 5\rangle$, segundo caminho mais curto de 3 para 5 diferente de $\langle 3, 5\rangle$ numa rede obtida da inicial removendo o nó $s = 1$. Iremos obter então o novo

Procedimento 6.5.3: *Procedimento CalcularPc(*$(\mathcal{N},\mathcal{A})$*,*$p_k$*)*
apagar nós de $\text{sub}_j(s, v^j_{\delta-1})$;
apagar arcos de $\mathcal{F}(v^j_\delta)$ de caminhos calculados a partir de p_j, que se desviam em v^j_δ;
$p \longleftarrow CMC((\mathcal{N},\mathcal{A}), v^j_\delta, t, \text{sub}_k(v^j_\delta, v^j_{\alpha_k}))$;
$P_c \longleftarrow \text{sub}_j(s, v^j_\delta) \diamond p$;
repor rede original;

Procedimento 6.5.4: *Procedimento CMC(*$(\mathcal{N},\mathcal{A})$*,*$s$*,*$t$*,*$sub_{p^\star}(s, v^\star_\alpha)$*)*

$\mathcal{T}_s \longleftarrow$ árvore dos trajectos mais curtos de s para todos os nós em $(\mathcal{N},\mathcal{A})$;
$\mathcal{T}_t \longleftarrow$ árvore dos trajectos mais curtos de todos os nós para t em $(\mathcal{N},\mathcal{A})$;
`Para` (todo o $u \in \mathcal{N}$) `Fazer`
  $\xi^s_u \longleftarrow$ nó que antecede u em $\mathcal{T}_s(u)$;
  $\beta^s_u \longleftarrow$ ordem do primeiro nó de $\mathcal{T}_s(u)$ que se separa de $p^\star$;
  $\beta^t_u \longleftarrow$ ordem do primeiro nó de $\mathcal{T}_t(u)$ que se separa de $p^\star$
`FimPara`
$d \longleftarrow +\infty$;
`Para` (todo o $u \in \mathcal{N}$ tal que $\beta^s_u < \alpha$) `Fazer`
  `Se` ($\beta^s_u = \beta^t_u$) `Então`
    `Para` (todo o $v \in \mathcal{D}(u)$ tal que $\xi^s_v \neq u$ e $\beta^s_u < \beta^t_v$) `Fazer`
      `Se` ($d > \pi^s_u + c_{uv} + \pi^t_v$) `Então` $p \longleftarrow \mathcal{T}_s(u) \diamond \langle u, v\rangle \diamond \mathcal{T}_t(v)$
    `FimPara`
  `FimSe`
  `Se` ($\beta^s_u < \beta^t_u$) `Então`
    `Para` (todo o $u \in \mathcal{N}$ tal que $\beta^s_u < \alpha$) `Fazer`
      `Se` ($d > \pi^s_u + \pi^t_u$) `Então` $p \longleftarrow \mathcal{T}_s(u) \diamond \mathcal{T}_t(u)$
    `FimPara`
  `FimSe`
`FimPara`

caminho $P_a = \langle 1, 3\rangle \diamond p = \langle 1, 3, 4, 5\rangle$.

Em seguida é determinado o caminho P_b, sendo para isso chamado o procedimento *CMC*$((\mathcal{N}', \mathcal{A}'), 1, 5, \langle 1, 2, 3, 5\rangle)$. Com esta chamada é calculado o caminho $p = \langle 1, 4, 3, 5\rangle$, segundo caminho mais curto de 1 para 5 diferente de p_1, numa rede $(\mathcal{N}', \mathcal{A}')$ obtida a partir de $(\mathcal{N}, \mathcal{A})$ removendo o primeiro arco de p_2, de modo a que este caminho não seja calculado novamente. Então $P_b = p$ e $P = \{\langle 1, 3, 4, 5\rangle, \langle 1, 4, 3, 5\rangle\}$. Uma vez que P_c não é determinado, no final deste passo a árvore dos caminhos de s para t é a representada na figura 32.

No passo seguinte é retirado de P o seu caminho com menor custo, $p_3 = \langle 1, 4, 3, 5\rangle$, cujo nó desvio é novamente $s = 1$, como acontecia para p_2. A partir da chamada de *CMC*$((\mathcal{N}', \mathcal{A}'), 4, 5, \langle 4, 3, 5\rangle)$, onde $(\mathcal{N}', \mathcal{A}')$ não contém o nó 1, é obtido $P_a = \langle 1, 4\rangle \diamond p$, com $p = \langle 4, 5\rangle$.

Em seguida, a partir da chamada do procedimento *CMC*$((\mathcal{N}', \mathcal{A}'), 1, 5, \langle 1, 2, 3, 5\rangle)$, é determinado o caminho P_b onde $(\mathcal{N}', \mathcal{A}')$ é uma rede obtida da original, onde foram removidos os arcos $(1, 3)$ e $(1, 4)$. Então $P_b = p$, onde $p = \langle 1, 2, 3, 4, 5\rangle$, é o novo caminho a acrescentar a P. Neste momento o conjunto P é constituído por

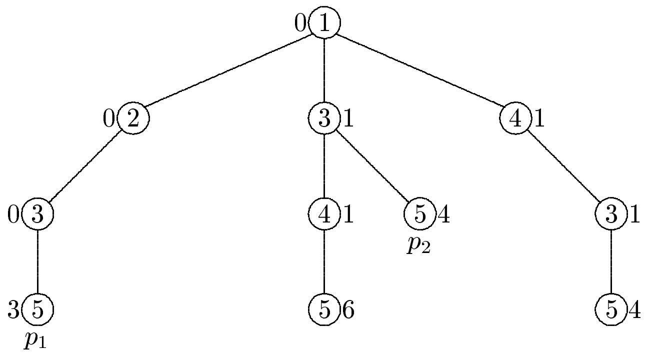

Figura 32

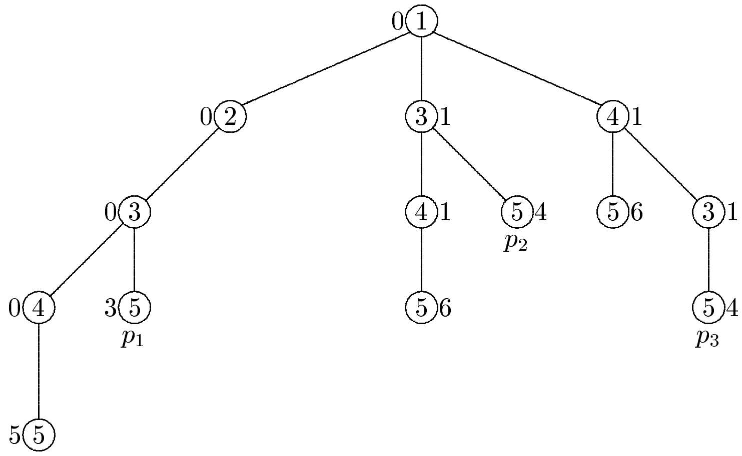

Figura 33

$P = \{\langle 1,3,4,5\rangle, \langle 1,4,5\rangle, \langle 1,2,3,4,5\rangle\}$.

Mais uma vez P_c não é determinado; no final da execução do segundo passo deste algoritmo, a árvore dos caminhos de s para t é a representada na figura 33.

No passo seguinte é retirado $p_4 = \langle 1,2,3,4,5\rangle$, o quarto caminho mais curto, do conjunto P. O nó desvio deste caminho é o seu nó de ordem 3, $v_3^4 = 3$. Na rede $(\mathcal{N}, \mathcal{A})$ não se definem os caminhos P_a e P_b verificando as condições indicadas. No entanto, p_4 foi calculado a partir de p_1 e $\langle 1,2,3\rangle$ é comum a estes dois caminhos, estando compreendido entre os respectivos nós desvio. Além disso, como $3 \in W_1$, isto é, como p_4 é o primeiro caminho calculado que se separa de p_1 no terceiro nó, então é calculado P_c. Como $\delta = \max\{\alpha \in W_1 : 0 < \alpha < 3\} = 1$, $P_c = p$, onde $p = \langle 1,2,5\rangle$ é o caminho resultante da chamada de $CMC((\mathcal{N}', \mathcal{A}'), 1, 5, \text{sub}_1(1,3))$. É de notar que a rede $(\mathcal{N}', \mathcal{A}')$ é obtida a partir da original, removendo os arcos $(1,3)$ e $(1,4)$, que pertencem aos caminhos p_2 e p_3, elementos da árvore determinada.

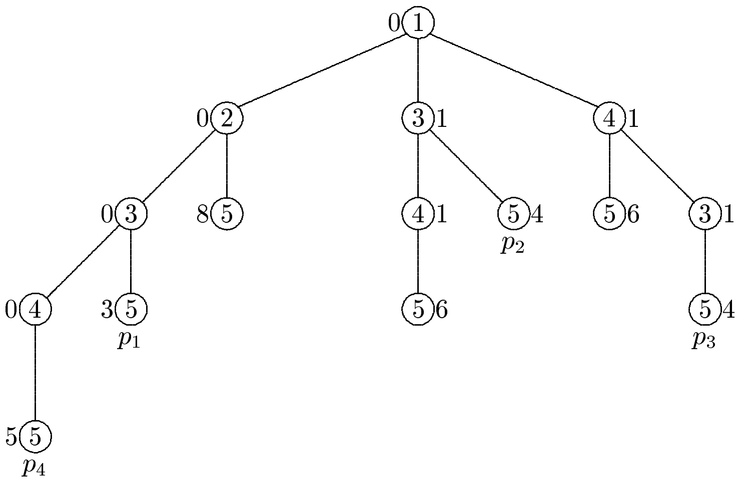

Figura 34

Assim, no final da execução deste passo a árvore dos caminhos é a representada na figura 34.

Em seguida é retirado de P o caminho $p_5 = \langle 1,3,4,5\rangle$, com nó desvio $v_2^5 = 3$. É chamado o procedimento $CMC((\mathcal{N}', \mathcal{A}'), 4, 5, \langle 4,5\rangle)$, onde $(\mathcal{N}', \mathcal{A}')$ não contém os nós 1 e 3. Com esta chamada não é calculado nenhum caminho, pelo que não se determina P_a.

A partir da chamada de $CMC((\mathcal{N}', \mathcal{A}'), 3, 5, \langle 3,5\rangle)$, onde $(\mathcal{N}', \mathcal{A}')$ não contém o arco $(3,4)$, é calculado $P_b = \langle 1,3\rangle \diamond p$, onde $p = \langle 3,2,5\rangle$. Apesar de p_5 ter o caminho $\langle 1,3\rangle$ em comum com p_2, $\langle 1,3\rangle$ contém apenas um arco, razão pela qual não é determinado P_c.

No final do quinto passo do algoritmo, $P = \{\langle 1,2,5\rangle, \langle 1,4,5\rangle, \langle 1,3,2,5\rangle\}$ e a árvore dos caminhos construída é a representada na figura 35.

Este algoritmo pode continuar a determinar novos caminhos, terminando quando for encontrado o $K^{\underline{\text{ésimo}}}$ caminho mais curto, ou p_9, uma vez que se definem apenas nove caminhos de 1 para 5 na rede em questão (ver figura 27, representando a árvore de todos os caminhos de 1 para 5 em $(\mathcal{N}, \mathcal{A})$).

7 Experiência Computacional

Nas secções anteriores foram descritos algoritmos para a resolução dos problemas dos K Trajectos e dos K Caminhos Mais Curtos numa rede, tendo sido estudadas as ordens de complexidade de alguns deles, considerando o pior dos casos. Os algoritmos apresentados podem ser comparados com base na respectiva ordem de

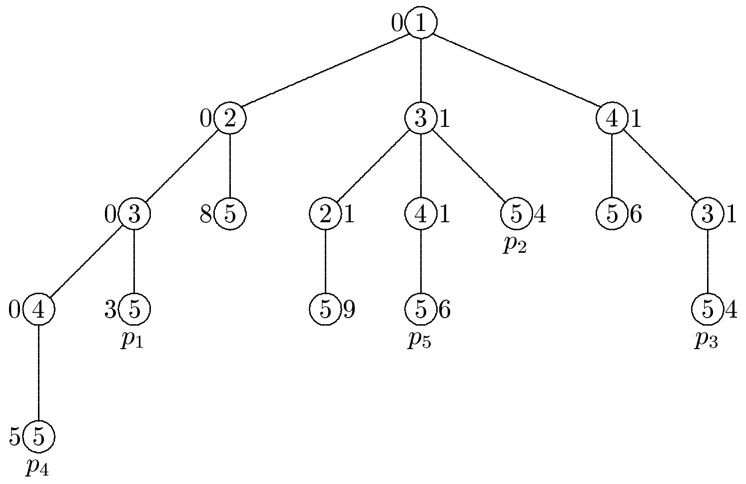

Figura 35

complexidade; no entanto podem ainda ser utilizados outros critérios de comparação, tais como o tempo de execução ou o espaço de memória que exige a resolução de um problema.

Nesta secção é apresentado um estudo dos tempos de execução e do espaço de memória ocupado, relativos a alguns dos algoritmos que foram descritos para a resolução dos problemas referidos.

Dos testes realizados foi possível seleccionar alguns algoritmos que se revelaram mais eficientes, ou seja, mais rápidos e ocupando menos memória, pelo que serão apresentados apenas resultados desses algoritmos.

Os códigos dos algoritmos testados, tanto para a ordenação de trajectos como para a ordenação de caminhos, foram escritos em linguagem C. Além disso são utilizadas rotinas idênticas para cálculos semelhantes, de modo a permitir uma comparação mais correcta. Outro factor que pode influenciar a rapidez de execução de cada algoritmo é a forma utilizada para representar uma rede, pelo que esta deve ser levada em conta aquando da implementação computacional de um algoritmo.

As experiências foram realizadas no servidor DEC AXP 7610 do Laboratório de Cálculo do Departamento de Matemática, com velocidade de processador de 275 MHz sobre DEC Unix 3.2. O servidor tem 128 Mbytes de RAM e 200.9SPECint92.

Esta secção é dividida em duas partes; a primeira é referente a algoritmos para a enumeração de trajectos, enquanto que a segunda é referente a algoritmos para a enumeração de caminhos.

7.1 Determinação Ordenada de Trajectos

Com base em alguns testes realizados concluímos que dos algoritmos que foram descritos na secção 5 para a resolução do problema dos K Trajectos Mais Curtos, o algoritmo de Eppstein (ou algoritmo EPP) e o algoritmo de Martins e Santos (ou algoritmo MS) estão entre os mais eficientes[6]. Assim, nesta subsecção será apresentado um breve estudo comparativo destes dois algoritmos.

Nos testes realizados foram utilizadas redes orientadas, tendo sido considerados dois tipos diferentes. No primeiro foram utilizadas redes geradas aleatoriamente, onde foi garantida a existência de pelo menos um trajecto do nó inicial para todos os outros, e de todos os nós para o terminal. Foi garantida ainda a não existência de arcos de um nó para ele próprio, e que entre um dado par ordenado de nós não existe mais do que um arco. As distâncias associadas a cada arco foram geradas aleatoriamente com distribuição uniforme, sendo inteiras, positivas e inferiores a um limite máximo dado. Foi ainda assegurado que os nós inicial e terminal fossem distintos.

No que respeita às dimensões, foram consideradas redes com 5000 nós e com densidades 4, 6, 8 e 10, onde a densidade de uma rede é o quociente entre o seu número de arcos e de nós, que denotaremos por $d = m/n$. Assim, o número de arcos considerado foi de 20000, 30000, 40000 e 50000. As distâncias máximas associadas a cada arco foram de 100, 1000 e 10000, para cada rede com um dado número de nós e uma certa densidade. O número K, de trajectos a calcular, foi de 100000. Para cada dimensão de rede foram resolvidos trinta problemas, tendo sido calculadas as médias dos tempos de execução de cada algoritmo.

Inicialmente foi escrito um código para o algoritmo de Eppstein que, de acordo com o algoritmo descrito na secção 5, guarda em memória todos os trajectos que são determinados. No entanto, isto dificulta a determinação de um elevado número de trajectos devido às limitações de memória. Com o objectivo de contornar esta limitação foi escrito um outro código para o mesmo algoritmo, em que apenas são guardados os K melhores desvios determinados em cada momento da execução. O aumento do número de operações necessárias neste código relativamente ao anterior originou um ligeiro aumento dos tempos de execução, permitindo no entanto a resolução de problemas de maiores dimensões. Os resultados apresentados são relativos ao segundo código escrito para o algoritmo de Eppstein.

Os tempos de execução obtidos para ambos os algoritmos são apresentados em várias figuras, começando-se pela figura 36 onde são consideradas as redes com 5000

[6] não foi ainda testado o algoritmo MPS para trajectos.

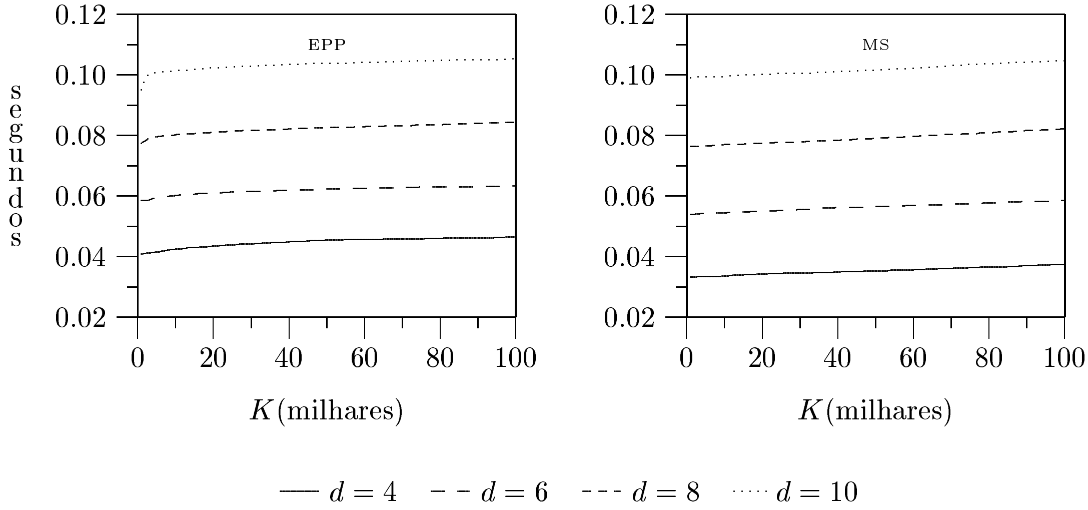

Figura 36 Tempos para redes aleatórias, distância máxima 100

nós e com arcos cuja distância máxima é 100.

Como se pode observar nesta figura, os tempos de execução obtidos para esta distância máxima são muito semelhantes para os dois algoritmos. No entanto, considerando distâncias até 1000, obtiveram-se os resultados apresentados na figura 37, onde é notória a maior rapidez do algoritmo de Martins e Santos (basta ter em atenção a escala dos tempos).

É ainda de notar que para redes com esta distância máxima, o tempo de execução do algoritmo de Martins e Santos aumenta de forma linear, quer segundo K quer segundo a densidade, ao passo que o tempo de execução do algoritmo de Eppstein cresce muito mais rapidamente, em especial segundo K.

Por fim, a figura 38 contém os tempos de execução obtidos pelos códigos dos algoritmos de Eppstein e do algoritmo de Martins e Santos para redes com arcos cuja distância máxima é 10000. A partir dos tempos apresentados nesta figura é possível confirmar a tendência verificada na figura 37, de um crescimento muito rápido dos tempos de execução do algoritmo de Eppstein, segundo K, e a linearidade para o algoritmo MS.

Da análise dos resultados obtidos para testes do primeiro tipo podemos pois concluir que o algoritmo de Eppstein é sensível à distância máxima dos arcos da rede. Este facto é justificado pela utilização do *adressed calculation sort* de Dijkstra, [6], para armazenar os trajectos calculados, no código escrito. Pelo contrário o código do algoritmo MS não é influenciado pela distância máxima dos arcos da rede, o que pode ser verificado apenas pela comparação das escalas dos gráficos das figuras 36,

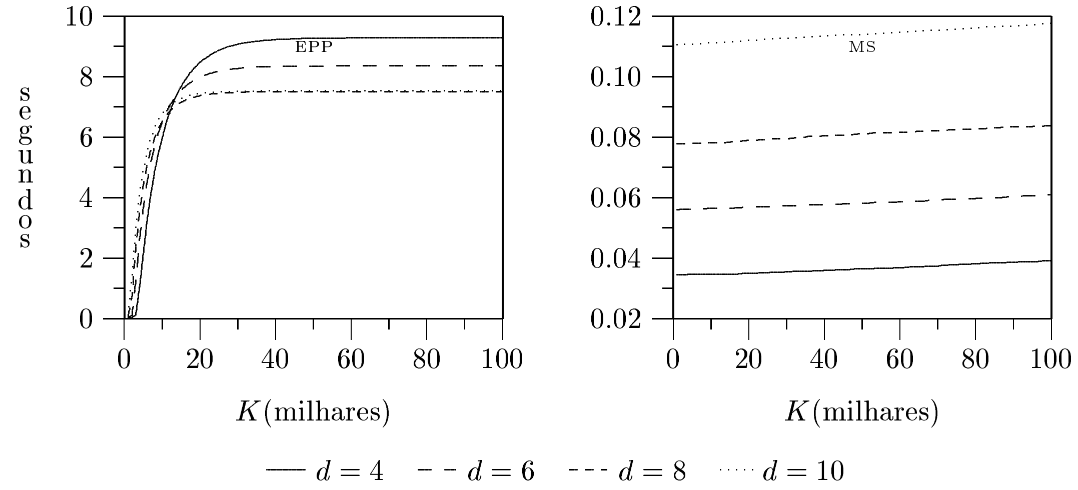

Figura 37 Tempos para redes aleatórias, distância máxima 1000

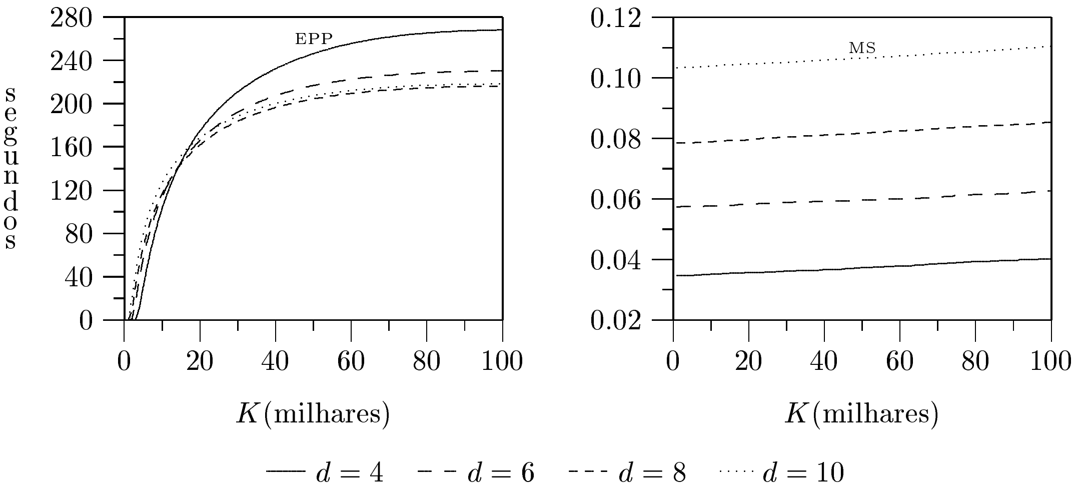

Figura 38 Tempos para redes aleatórias, distância máxima 10000

37 e 38.

Como referimos, podemos igualmente concluir que o algoritmo MS apresenta um tempo de execução dependente quer do número de trajectos calculados quer da densidade. De facto o aumento da densidade da rede induz o aumento do tempo de execução de ambos os algoritmos, uma vez que implica um aumento do número de arcos da rede.

Ainda com redes aleatórias com 10000 nós, foi testado outro problema de maiores dimensões. O principal objectivo destes testes foi ter uma ideia do número máximo de trajectos que era possível determinar, para um certo espaço de memória disponível. Neste caso a memória disponível foi fixada em 115Mb de RAM e foram resolvidos cinco problemas com cinco redes de densidades distintas. Nestes testes utilizámos apenas o algoritmo MS uma vez que, como observámos nas figuras 36, 37 e 38, o algoritmo de Eppstein exige um tempo de execução demasiado elevado.

Os resultados obtidos encontram-se resumidos na tabela 5.

		Tempo (seg)		N	
d	último K	ST	Total	Média	Máximo
2	435089	0.05	0.15	38.32	59
4	647185	0.09	0.32	25.88	49
6	692271	0.09	0.22	22.18	42
8	730264	0.17	0.35	22.49	47
10	637103	0.20	0.29	23.48	44

N número de nós dos trajectos
último K Número de trajectos calculados
ST Tempo para calcular a árvore dos trajectos $\mathcal{T}_s$
Total Tempo para calcular todos os trajectos

Tabela 5 Resultados para redes aleatórias com 10000 nós

Destes resultados é interessante observar que grande parte do tempo total de execução do algoritmo se deve à determinação da árvore dos trajectos de s para todos os nós, realizada uma única vez, o que reafirma a rapidez do algoritmo. Por outro lado, em redes com densidades menores foram determinados menos trajectos. Isto pode ser justificado pelo facto de o número médio de nós por trajecto ser maior para densidades pequenas, o que implica, de acordo com o algoritmo, que seja necessário realizar um maior número de cópias de nós para cada trajecto.

Foram ainda realizados testes com redes do tipo indicado na figura 39 e designadas por redes do tipo **grelha** ou simplesmente grelhas. A grelha daquela figura tem a nós de altura e l de largura, pelo que o seu número de nós é $n = a \times l$.

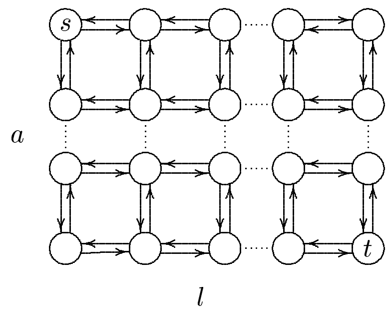

Figura 39

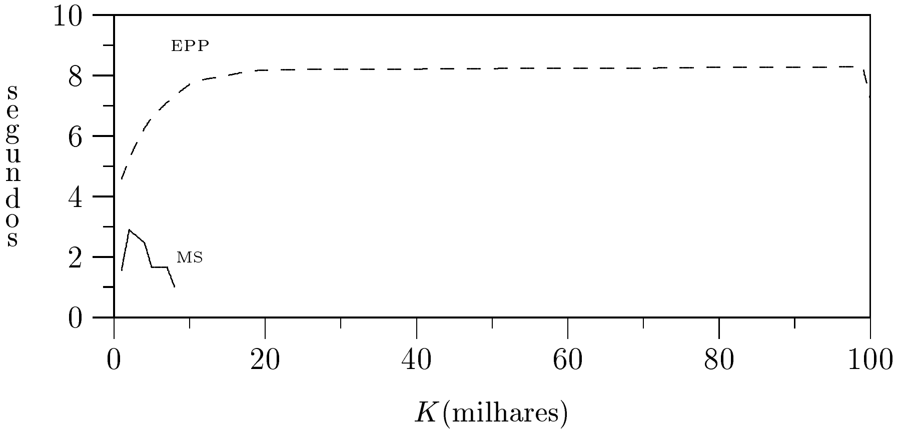

Figura 40 Tempos para grelhas 6×864, distância máxima 1000

Quando é gerada uma rede deste tipo é associado aleatoriamente a cada arco um valor inteiro positivo que representa a respectiva distância.

Dos testes efectuados para o algoritmo MS obtiveram-se alguns tempos de execução para redes do tipo grelha; no entanto, e como se pode observar no gráfico da figura 40, muitos problemas ficaram por resolver, não se tendo obtido tempos que permitissem tirar conclusões. Isto deve-se ao elevado espaço de memória exigido por este algoritmo à medida que são construídas as redes $(\mathcal{N}_k, \mathcal{A}_k)$, com $k \geq 1$. De facto, o problema dos K Trajectos Mais Curtos é de mais difícil resolução em redes do tipo grelha, uma vez que, em geral, os trajectos nestas redes são constituídos por um elevado número médio de nós. Contudo o algoritmo de Eppstein, ao contrário do que acontecia para os tipos de redes anteriores, resolveu a maior parte dos problemas propostos, embora não muito rapidamente.

Apresentamos, na figura 40, alguns dos resultados obtidos para uma grelha onde $a = 864$ e $l = 6$, contendo 5184 nós e 18996 arcos, cuja distância máxima dos arcos foi fixada em 1000.

7.2 Determinação Ordenada de Caminhos

Nesta subsecção será apresentado um estudo comparativo dos algoritmos de Yen, de Martins, Pascoal e Santos (algoritmo MPS) e de Katoh, Ibaraki e Mine (algoritmo KIM) para a resolução do problema dos K Caminhos Mais Curtos, descritos na secção 6.

Nas experiências computacionais realizadas foram utilizadas somente redes não orientadas, uma vez que o algoritmo de KIM é válido apenas para este tipo de redes.

De entre as redes não orientadas foram considerados dois tipos diferentes. O primeiro inclui **redes euclidianas**, isto é, redes onde cada nó é considerado como sendo um ponto do plano, com coordenadas inteiras e positivas, geradas aleatoriamente dentro de um intervalo de variação imposto previamente. Cada arco é determinado aleatoriamente e a distância que lhe está associada corresponde à distância euclidiana entre os nós que o definem, arredondada por forma a obter apenas valores inteiros. Mais uma vez foi garantida a existência de pelo menos um trajecto de s para todos os outros nós da rede, e de todos esses nós para t, não existindo arcos de um nó para ele próprio e existindo no máximo um arco entre cada par ordenado de nós. Foi ainda garantido que $s \neq t$.

Foram utilizadas redes de várias dimensões, tendo sido resolvidos dez problemas para a mesma dimensão e calculadas as médias dos tempos de execução. No entanto, devido à semelhança dos resultados obtidos, apresentamos somente os tempos de execução dos três algoritmos para redes com 10000 nós, de densidades 2 e 10, determinando $K = 1000$ caminhos (ver figuras 41 e 42).

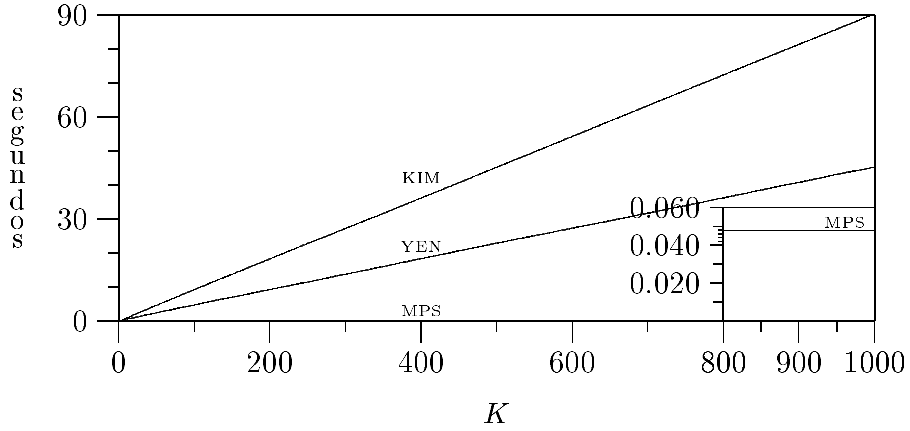

Figura 41 Tempos para redes euclidianas, $n = 10000, d = 2$, *zoom* para $K \geq 800$

No segundo grupo de testes foram utilizadas redes do tipo grelha, de dimensão $a \times 25$, onde as alturas consideradas foram de 10, 20, 25, 30, 40, 50 e 100. Nova-

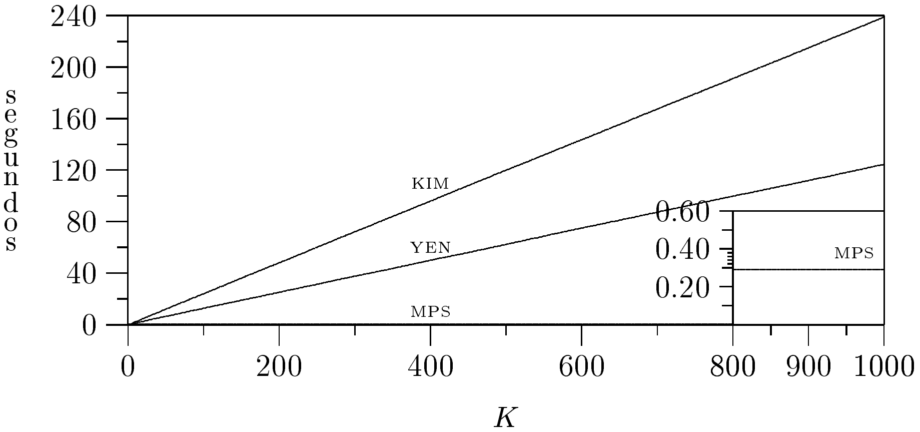

Figura 42 Tempos para redes euclidianas, $n = 10000, d = 10$, *zoom* para $K \geq 800$

mente, para cada dimensão foram resolvidos dez problemas determinando $K = 1000$ caminhos, sendo em seguida calculadas as médias dos tempos de execução de cada algoritmo. Na figura 43 apresentam-se os tempos de execução obtidos pelos três algoritmos para grelhas com uma altura de 100 nós.

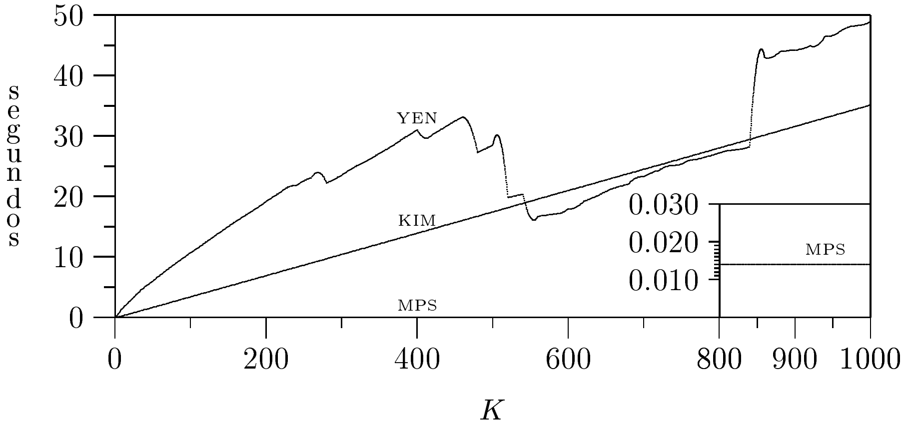

Figura 43 Tempos para grelhas 100×25, *zoom* para $K \geq 800$

É de notar que para estes problemas, o algoritmo MPS apresentou tempos muito inferiores aos dos outros dois, sendo mesmo constantes, enquanto que os outros dois apresentaram tempos com um comportamento aparentemente linear, segundo K.

O espaço de RAM necessário não é o mesmo para todos os códigos, sendo o algoritmo de KIM aquele que necessita de menos memória, seguido do algoritmo MPS. Dos tempos apresentados para grelhas 100×25 e $K = 1000$ é de notar que alguns dos problemas propostos não foram resolvidos pelo algoritmo de Yen, o que justifica o gráfico da figura 43.

Dado o facto de os tempos obtidos para o algoritmo MPS serem praticamente constantes e de modo a testá-lo mais eficazmente, este algoritmo foi aplicado a problemas com redes de maiores dimensões e determinando um maior número de caminhos. Tal como foi realizado para trajectos com o algoritmo MS, testou-se o número máximo de caminhos que foi possível determinar, o espaço de RAM ocupado e o tempo dispendido na resolução de um problema. Nestes testes foram utilizadas redes euclidianas, grelhas e redes completas, quer orientadas quer não orientadas, sendo resolvidos cinco problemas para cada dimensão da rede.

Os resultados obtidos são resumidos na tabela 6.

	$K=1$		$K=100000$		último K				T.T.	
	segundos				K		segundos		segundos	
	n.ori.	ori.	n.ori.	ori.	n.ori.	ori.	n.ori.	ori.	n.ori.	ori.
completa $n=1000$	1.019	1.098	2.035	2.127	609099	742717	2.042	2.135	0.008	0.009
	0.976	0.859	2.006	1.880	546775	837756	2.014	1.887	0.010	0.008
	1.000	0.991	2.034	2.017	650736	828785	2.042	2.024	0.010	0.008
	0.965	1.077	1.899	1.969	521609	739092	1.907	1.976	0.010	0.008
	1.016	0.941	2.042	1.976	578996	808513	2.051	1.985	0.011	0.010
média	1.008	0.978	2.003	1.994	581443.	783329.	2.011	2.001	0.010	0.009
euclidiana $n=10000$ $d=10$	0.166	0.068	0.324	0.130	599193	621475	0.334	0.144	0.012	0.016
	0.149	0.066	0.299	0.130	574896	646515	0.309	0.145	0.012	0.016
	0.184	0.073	0.335	0.133	441301	603677	0.346	0.148	0.014	0.016
	0.173	0.071	0.322	0.129	526221	630502	0.334	0.146	0.014	0.017
	0.166	0.064	0.316	0.125	530606	610426	0.327	0.138	0.013	0.014
média	0.168	0.068	0.319	0.129	534443.	622519.	0.330	0.144	0.013	0.016
grelha 1000×25	2.034	2.005	n.c.	n.c.	30275	10743	2.154	2.216	0.075	0.164
	1.962	2.170	n.c.	n.c.	14441	13591	2.055	2.283	0.081	0.065
	2.122	2.183	n.c.	n.c.	16636	49314	2.238	2.316	0.068	0.084
	1.736	1.923	n.c.	n.c.	23579	44671	1.873	2.046	0.090	0.075
	1.878	2.225	n.c.	n.c.	31996	17557	2.001	2.329	0.077	0.055
média	1.946	2.101	——	——	23385.	27175.	2.064	2.238	0.078	0.089
grelha 150×150	0.156	0.168	0.211	0.217	107044	210464	0.213	0.225	0.016	0.015
	0.154	0.152	n.c.	n.c.	92535	50832	0.216	0.204	0.021	0.010
	0.141	0.176	n.c.	n.c.	97133	98567	0.197	0.235	0.014	0.015
	0.210	0.167	n.c.	n.c.	85051	80173	0.266	0.220	0.014	0.011
	0.173	0.179	n.c.	n.c.	87537	68167	0.225	0.231	0.010	0.009
média	0.167	0.168	——	——	93862.	101641.	0.223	0.223	0.015	0.012

n.ori. – redes não orientadas; ori. – redes orientadas; n.c. – não correu
T.T. – Tempo total $-(\mathcal{T}_t +$ mudança de variável $+$ ordenação dos arcos)

Tabela 6 Resultados do algoritmo MPS para redes de grandes dimensões

É de realçar que os problemas com redes completas usaram cerca de 125 Mb de RAM, ao passo que os problemas com redes euclidianas e grelhas usaram cerca de 105 Mb de RAM.

De acordo com a tabela 6 e com os resultados relativos ao tempo de execução, o facto de uma rede ser ou não orientada parece ser importante apenas para redes euclidianas; no entanto, tendo em conta o número máximo de caminhos determinados, os problemas parecem mais simples em redes orientadas, embora tal não seja evidente para redes do tipo grelha.

Tal como havíamos concluído para trajectos, também a ordenação de caminhos em redes do tipo grelha parece ser mais morosa do que em redes de outros tipos.

A densidade da rede também parece influenciar o tempo de execução, o que pode justificar os tempos apresentados para redes completas, implicando um aumento do tempo dispendido para a determinação da árvore dos trajectos mais curtos com raiz em t, bem como da ordenação dos arcos da rede.

Resumindo, dos resultados apresentados podemos afirmar que o algoritmo MS se revelou bastante eficiente para a ordenação de trajectos, embora a sua última versão, até ao momento, ocupe ainda bastante espaço de memória, em especial quando os trajectos determinados contêm um elevado número de nós.

Por outro lado, o algoritmo MPS mostrou-se bastante eficiente para a ordenação de caminhos, sendo bastante mais rápido do que o algoritmo de Yen e do algoritmo KIM, apresentando tempos de execução quase constantes relativamente ao número de caminhos determinados numa rede.

8 Conclusão

Neste trabalho pretendemos apresentar um estudo sobre o problema dos K Trajectos Mais Curtos na sua vertente algorítmica, não esquecendo o necessário suporte teórico.

Foi realizada uma classificação dos algoritmos conhecidos até ao início do estudo levado a efeito. Deste estudo resultou a consideração de dois tipos de problemas, dependentes da existência ou não de restrições na definição de trajecto. Feita a classificação dos algoritmos para o problema sem restrições, estes foram divididos em dois grupos: os que se baseiam no Princípio de Optimalidade e os que não têm este princípio em consideração.

De entre os problemas com restrições, foi estudado apenas o da determinação dos K caminhos mais curtos – geralmente considerado de grande dificuldade – que não satisfaz o dito Princípio de Optimalidade. O algoritmo existente na altura do início deste estudo, deve-se a Yen e é, obviamente, um algoritmo que não se baseia neste Princípio. Foi constatado que a sua adaptação para o problema sem restrições é possível, assim como, e de um modo semelhante, a adaptação dos algoritmos para o problema sem restrições (que não se baseiam no Princípio de Optimalidade) para a determinação dos K caminhos mais curtos. Surgem deste modo alguns algoritmos, que são apresentados desde o menos elaborado (e menos eficiente) até ao mais elaborado (e mais eficiente).

Como trabalho a desenvolver parecem-nos ser de considerar, entre outros possíveis, os seguintes pontos:

- Testar mais aprofundadamente, do ponto de vista computacional, os algoritmos apresentados.

- Estudar a viabilidade da adaptação dos algoritmos que se concluam ser mais eficientes, para outro tipo de problemas da determinação de K trajectos com restrições.

- Estudar a adaptação dos algoritmos a tipos específicos de redes. Nomeadamente, redes não orientadas (será que o método usado no algoritmo de KIM pode ser usado noutros algoritmos?) e redes acíclicas (qual dos algoritmos – MS, MPS, de Eppstein e GY – é mais eficiente?).

- A aplicabilidade do Problema dos K Trajectos Mais Curtos na resolução de Problemas do Trajecto Óptimo e, eventualmente, noutro tipo de problemas.

Lista de Algoritmos

Lista de Figuras

Lista de Tabelas

Referências

[1] J.A. Azevedo, M.E.O.S. Costa, J.J.E.R.S. Madeira e E.Q.V. Martins, An algorithm for the ranking of shortest paths, European Journal of Operational Research 69, (1993), 97–106.

[2] J.A. Azevedo, J.J.E.R.S. Madeira, E.Q.V. Martins e F.M.A. Pires, A shortest paths ranking algorithm, (1990), Proceedings of the Annual Conference AIRO'90, Models and Methods for Decision Support, Operational Research Society of Italy, 1001-1011.

[3] J.A. Azevedo, J.J.E.R.S. Madeira, E.Q.V. Martins e F.M.A. Pires, A computational improvement for a shortest paths ranking algorithm, European Journal of Operational Research 73, (1994), 188–191.

[4] R.E. Bellman, On a routing problem, Quarterly Applied Mathematics, (1958), 1:425–447.

[5] B.V. Cherkassky, A.V. Goldberg e T. Radzik, Shorthest paths algorithms: Theory and experimental evaluation, Mathematical Programming 73, (1996), 129–196.

[6] R. Dial, G. Glover, D. Karney e D. Klingman, A computational analysis of alternative algorithms and labelling techniques for finding shortest path trees, Networks 9, (1979), 215–348.

[7] E. Dijkstra, A note on two problems in connection with graphs, Numerical Mathematics 1, (1959), 395–412.

[8] S.E. Dreyfus, An appraisal of some shortest-path algorithms, Operations Research 17, (1969), 395–412.

[9] D. Eppstein, Finding the k shortest paths, SIAM Journal on Computing (aceite para publicação).

[10] G. Gallo e S. Pallotino, Shortest path methods: a unifying approach, Mathematical Programming Study 26, (1986), 38–64.

[11] D. Hochbaum, Graph algorithms and network flows, Apontamentos de aulas proferidas na Universidade da Califórnia, (1997), 21–22.

[12] R. Hoffman e R. R. Pavley, A method for the solution of the N th best path problem, Journal of the Association for Computing Machinery 6, (1959), 506–514.

[13] N. Katoh, T. Ibaraki e H. Mine, An efficient algorithm for K shortest simple paths, Networks 12, (1982), 411–427.

[14] E.Q.V. Martins, An algorithm for ranking paths that may contain cycles, European Journal of Operational Research 18, (1984), 123–130.

[15] E.Q.V. Martins, **Determinação de caminhos óptimos em redes orientadas**, Dissertação de Doutoramento, (1984), Departamento de Matemática, Universidade de Coimbra.

[16] E.Q.V. Martins, Sobre quatro problemas do melhor caminho, Actas do 1º Ciclo de Conferências em Análise Numérica e Optimização, (1986), Departamento de Matemática, Universidade de Coimbra, 24–38.

[17] E.Q.V. Martins e J.L.E. Santos, An new shortest paths ranking algorithm, submetido.

[18] E.Q.V. Martins, M.M.B. Pascoal e J.L.E. Santos, A new algorithm for ranking loopless paths, submetido.

[19] E.F. Moore, The shortest path through a maze, In Proc. of the Int. Symp. on the Theory of Switching, (1959), Harvard University Press, 285–292.

[20] U. Pape, Implementation and efficiency of Moore algorihms for the shortest root problem, Mathematical Programming. 7, (1974), 212–222.

[21] A. Perko, Implementation of algorihms for K shortest loopless paths, Networks 16, (1986), 149–160.

[22] D. Shier, Computational experience with an algorithm for finding the K shortest paths in a network, Journal of Research of the NBS 78 B Nº 3, (1974), 139–164.

[23] D. Shier, Interactive methods for determining the k shortest paths in a network, Networks 6, (1976), 151–159.

[24] D. Shier, On algorithms for finding the K shortest paths in a network, Networks 9, (1979), 195–214.

[25] J.W. Suurballe, Disjoint paths in a network, Networks 4, (1974), 125–145.

[26] J.Y. Yen, Finding the K shortest loopless paths in a network, Management Science 17, (1971), 712–716.

[27] J.Y. Yen, Shortest path network problems, (Mathematical Systems in Economics, Heft 18), Hain, (1975), Meisennheim am Glan.