OPTIMIZAÇÃO EM REDES

UMA VISÃO GLOBAL

Deolinda Dias Rasteiro, Jose Luis Santos, Marta Braz Pascoal

{*dml, zeluis, marta*}*@mat.uc.pt*

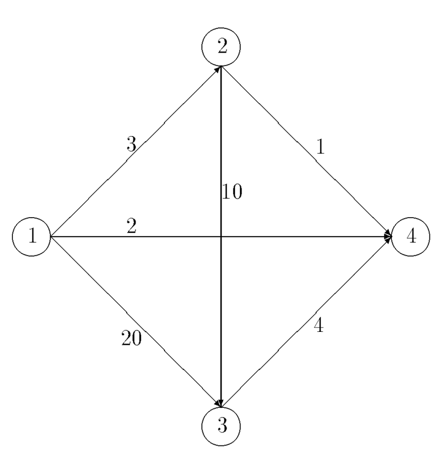
Departamento de Matemática, Universidade de Coimbra

Braga, Abril de 1999

Os autores pertencem à Linha *Ciências da Computação* do Centro de Informática e Sistemas da Universidade de Coimbra, parcialmente subsidiado pelo Ministério das Ciências e Tecnologia através do Projecto PRAXIS XXI da Junta Nacional de Investigação Científica e Tecnológica.

$\mathcal{N} = \{v_1, \ldots, v_n\} \equiv \{1, \ldots, n\}$	-	conjunto de nós (vértices)
$\mathcal{A} = \{a_1, \ldots, a_m\} \subset \mathcal{N} \times \mathcal{N}$	-	conjunto de arcos (arestas)
c_{ij}	-	distância associada ao arco (i, j)
Rede $(\mathcal{N}, \mathcal{A})$	-	grafo $(\mathcal{N}, \mathcal{A})$ com um vértice inicial s e um vértice terminal t, a cujos arcos estão associados parâmetros

Exemplo:



Representação de uma Rede:

- **Matriz de incidência vértice-vértice:**

$$M \equiv [a_{ij}]_{\substack{i=1,\ldots,n \\ j=1,\ldots,n}} = \begin{cases} 1 & \text{, se existe a aresta } (i,j) \\ 0 & \text{, caso contrário} \end{cases}$$

Exemplo:

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- **Matriz de incidência vértice-arco:**

$$M \equiv [a_{ia_k}]_{\substack{i=1,\ldots,n \\ k=1,\ldots,m}} = \begin{cases} 1 & \text{, se o arco } a_k \text{ se inicia em } i \\ -1 & \text{, se o arco } a_k \text{ termina em } i \\ 0 & \text{, caso contrário} \end{cases}$$

Exemplo:

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccc} (1,2) & (1,3) & (1,4) & (2,3) & (2,4) & (3,4) \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix} \end{array}$$

• **Dois Vectores** $+ \cdots$**:**

ANT – vector dos nós antecessores;
SUC – vector dos nós sucessores;
$\cdots$ – os vectores necessários para armazenar os parâmetros associados aos arcos;

$$a_k \in \mathcal{A} \text{ e } a_k = (i, j) \text{ então}$$

$$\mathbf{ANT}(k) = i \;\; ; \;\; \mathbf{SUC}(k) = j \;\; ; \;\; \mathbf{DIST}(k) = c_{ij}$$

Exemplo:

ARCOS	ANT	SUC	DIST
1	1	2	3
2	1	3	20
3	1	4	2
4	2	3	10
5	2	4	1
6	3	4	4

- **FORWARD STAR FORM**[1]**:**

POINT	–	vector de dimensão $n+1$ ($n = \sharp\, \mathcal{N}$);
SUC	–	vector dos nós sucessores;
$\cdots$	–	vectores necessários para armazenar os parâmetros associados aos arcos;

1) Numeram-se os arcos por forma a que

$$a_k = (i,j) < a_l = (i',j') \Leftrightarrow i \leq i' \;\; \forall a_k, a_l \in \mathcal{A}$$

2) **POINT**$(1) = 1$;

POINT$(i) =$ **POINT**$(i-1) + n^{\underline{o}}$ de arcos que se iniciam em $(i-1)$, $\forall i = 2, \ldots, n$;

POINT$(n+1) = m+1$;

3) Constrói-se o vector **SUC** mais $\cdots$.

Exemplo:

POINT $=$ $\boxed{1 \mid 4 \mid 6 \mid 7 \mid 7}$

SUC $=$ $\boxed{2 \mid 3 \mid 4 \mid 3 \mid 4 \mid 4}$

DIST $=$ $\boxed{3 \mid 20 \mid 2 \mid 10 \mid 1 \mid 4}$

[1] Considera-se sempre que a rede é orientada.

- **REVERSE STAR FORM**[2]**:**

RPOINT	–	vector de dimensão $n+1$ ($n = \sharp\,\mathcal{N}$);
ANT	–	vector dos nós antecessores;
$\cdots$	–	vectores necessários para armazenar os parâmetros associados aos arcos;

1) Numeram-se os arcos por forma a que

$$a_k = (i,j) < a_l = (i',j') \Leftrightarrow j \leq j' \;\; \forall a_k, a_l \in \mathcal{A}$$

2) **RPOINT**$(1) = 1$;
RPOINT(i) = **RPOINT**$(i-1) + n^{\underline{o}}$ de arcos que terminam em $(i-1)$, $\forall i = 2, \ldots, n$;
RPOINT$(n+1) = m+1$;

3) Constrói-se o vector **ANT** mais $\cdots$.

Exemplo:

$$\textbf{RPOINT} = \boxed{1}\boxed{1}\boxed{2}\boxed{4}\boxed{7}$$

$$\textbf{ANT} = \boxed{1}\boxed{1}\boxed{2}\boxed{1}\boxed{2}\boxed{3}$$

$$\textbf{DIST} = \boxed{3}\boxed{20}\boxed{10}\boxed{2}\boxed{1}\boxed{4}$$

[3]Considera-se sempre que a rede é orientada.

Dados $v_i, v_j \in \mathcal{N}$

- uma **cadeia** de v_i para v_j é uma sequência da forma:

$$\langle v_i \equiv v_0, a_1, v_1, \ldots, a_l, v_l \equiv v_j \rangle$$

  onde

  - $l \geq 1$;
  - $v_k \in \mathcal{N}$, para $k = 0, \ldots, l$;
  - $a_k \in \mathcal{A}$, para $k = 1, \ldots, l$;
  - para $k = 1, \ldots, l$ se a_k é orientado
    então $a_k = (v_{k-1}, v_k)$ ou $a_k = (v_k, v_{k-1})$
    senão $a_k = (v_{k-1}, v_k)$;

- um **circuito** de v_i para v_j é uma cadeia onde $v_i \equiv v_j$;
- um **trajecto** de v_i para v_j é uma cadeia de v_i para v_j onde os arcos são percorridos no sentido correcto[3];
- um **caminho** de v_i para v_j é um trajecto de v_i para v_j onde não são permitidas repetições de vértices com excepção do inicial e do terminal que podem coincidir [**ciclo**].

Um grafo diz-se **conexo** se existir, pelo menos, uma cadeia entre quaisquer dos seus nós.

Uma **árvore** é um grafo conexo e sem circuitos. Dado um nó s de $(\mathcal{N}, \mathcal{A})$ uma **árvore** diz-se ter **raiz em** s se é uma árvore onde existe um (e um só) trajecto de s para qualquer outro nó;

Uma **árvore de cobertura de** $(\mathcal{N}, \mathcal{A})$ é um subgrafo de $(\mathcal{N}, \mathcal{A})$ que é uma árvore e cujos nós são o conjunto $\mathcal{N}$.

[6] Num grafo não orientado a noção de trajecto e cadeia são equivalentes.

Problema do Trajecto Óptimo:

Dados:

- Rede $(\mathcal{N}, \mathcal{A})$, ($\mathcal{N} = \{\text{nós}\}$, $\mathcal{A} = \{arcos\}$)
- s, t dois vértices (geralmente diferentes)
- $c : \mathcal{P} \longrightarrow \mathbb{R} : p \longrightarrow c(p)$ ($\mathcal{P} = \{\text{trajectos de } s \text{ para } t\}$)
- $\mathcal{P}^\star \subseteq \mathcal{P}$

Pretende-se:

- Determinar $p^\star \in \mathcal{P}^\star$ tal que $c(p^\star) \leq c(p)$ $(c(p^\star) \geq c(p))$, qualquer que seja $p \in \mathcal{P}^\star$.

$\leq$ *ou* $\geq$ depende do tipo de optimização pretendida, (minimização ou maximização, respectivamente).

Alguns Problemas do Trajecto Óptimo

Exemplos:	**(1)**	$c(p)$	**Objectivo**
Trajecto mais Curto	$c_{ij} \in \mathbb{R}$	$c(p) = \sum_p c_{ij}$	$\min_{\mathcal{P}} c(p)$
Caminho mais Curto	$c_{ij} \in \mathbb{R}$	$c(p) = \sum_p c_{ij}$	$\min_{\mathcal{P}^\star} c(p)$ $\mathcal{P}^\star = \{\text{caminhos de } s \text{ para } t\}$
Trajecto de Capacidade Máxima	$u_{ij} \in \mathbb{R}^+$	$c(p) = \min_p\{u_{ij}\}$	$\max_{\mathcal{P}} c(p)$
Trajecto de Menor Custo por Unidade de Capacidade	$c_{ij} \in \mathbb{R}_0^+$ $u_{ij} \in \mathbb{R}^+$	$c(p) = \frac{\sum_p c_{ij}}{\min_p\{u_{ij}\}}$	$\min_{\mathcal{P}} c(p)$
Caixeiro Viajante	$c_{ij} \in \mathbb{R}$	$c(p) = \sum_p c_{ij}$	$\min_{\mathcal{C}^\star} c(p)$ $\mathcal{C}^\star = \{\text{ciclos hamiltoneanos}\}$

Problemas Especiais:

- Enumeração de Trajectos.
- Trajecto Óptimo Vectorial ($c : \mathcal{P} \longrightarrow \mathbb{R}^k,\ k > 1$).
- Trajecto Óptimo Estocástico.

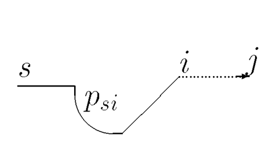
(1) – Parâmetros associados ao arco (i, j)

Princípio de Optimalidade

FORTE — **Todo** o trajecto óptimo de s para t é constituído por subtrajectos óptimos de s para qualquer vértice intermédio i.

FRACA — **Existe um** trajecto óptimo de s para t que é constituído por subtrajectos óptimos de s para qualquer vértice intermédio i.

O Princípio da Optimalidade Fraca suporta uma classe de algoritmos — **Algoritmos de Rotulação.**



$$p_{si} \diamond \langle i, j \rangle$$

Concatenação de p_{si} com $\langle i, (i, j), j \rangle$

ALGORITMO GERAL DE ROTULAÇÃO

Minimização

$X \longleftarrow \{s\}$
$p_{si} \longleftarrow \emptyset$ ($c(p_{si}) \longleftarrow +\infty$), para todo $i \in \mathcal{N} - \{s\}$;
$c(p_{ss}) \longleftarrow 0$
Enquanto $X \neq \emptyset$ **faz**:
    $i \longleftarrow$ vértice de X
    Para todo $j \in \mathcal{N}$ **tal que** $(i,j) \in \mathcal{A}$ **faz:**
        Se $c(p_{si} \diamond \langle i,j \rangle) < c(p_{sj})$ **então:**
            $p_{sj} \longleftarrow p_{si} \diamond \langle i,j \rangle$
            Se $j \notin X$ **então** $X \longleftarrow X \cup \{j\}$
    $X \longleftarrow X - \{i\}$

NOTA: Não é imposta qualquer regra quer para a manipulação de X quer para a escolha de i nesse conjunto.

Algoritmos de Rotulação	Rótulos não Definitivos	Bellman–Ford–Moore Pape–Levit ...
	Rótulos Definitivos — Algoritmo de Dijkstra	

Rótulos Definitivos — $i \in \mathcal{N}$ é escolhido em X por forma a que p_{si} seja óptimo.

Bellman–Ford–Moore — X é uma lista **FIFO**.

Pape–Levit — X é uma lista **FIFO** ou **LIFO**. (Remoções sempre no início e inserção de j no início se e só se j nunca pertenceu a X).

O **Problema do Trajecto mais Curto** verifica os Princípios de Optimalidade $\iff \sum_{C} c_{ij} \geq 0$, para todo o ciclo C da rede.

$\sum_{C} c_{ij} \geq 0$, para todo o ciclo C da rede $\Longrightarrow$ **Problema do Caminho mais Curto** verifica os Princípios de Optimalidade.

O **Problema do Trajecto de Capacidade Máxima** não verifica em geral o Princípio de Optimalidade Forte

verifica o Princípio de Optimalidade Fraca.

DOIS PROBLEMAS DO TRAJECTO ÓPTIMO

Dados:

- $s,\ t \Leftarrow$ nó inicial e nó terminal
- $(i,j) \Leftarrow c_{ij} \in \mathbb{R},\ u_{ij} \in \mathbb{R}^+$
- $c : \mathcal{P} \longrightarrow \mathbb{R} : p \longrightarrow c(p) = \sum_{p} c_{ij}$
- $u : \mathcal{P} \longrightarrow \mathbb{R} : p \longrightarrow u(p) = \min_{p} u_{ij}$
- $\mathcal{P}^* = \{$trajectos mais curtos em $\mathcal{P}\}$
- $\overline{\mathcal{P}} = \{$trajectos de capacidade máxima em $\mathcal{P}\}$

Pretende-se determinar:

1. O trajecto de capacidade máxima no conjunto dos trajectos mais curtos

$$(p^* \in \mathcal{P}^* \textbf{ tal que } u(p^*) \geq u(p),\ \ \forall\ p \in \mathcal{P}^*)$$

Satisfaz o **princípio de optimalidade fraca**

2. O trajecto mais curto no conjunto dos trajectos de capacidade máxima

$$(\overline{p} \in \overline{\mathcal{P}} \textbf{ tal que } c(\overline{p}) \leq c(p),\ \ \forall\ p \in \overline{\mathcal{P}})$$

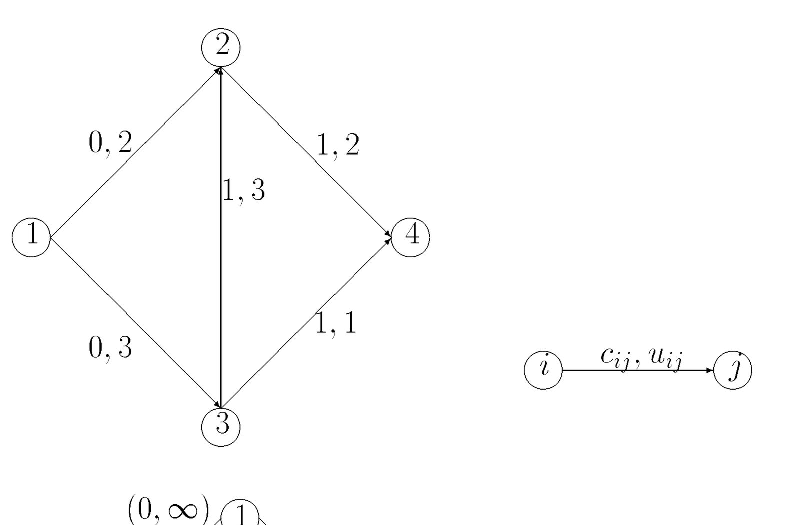
Não satisfaz o **princípio de optimalidade fraca**

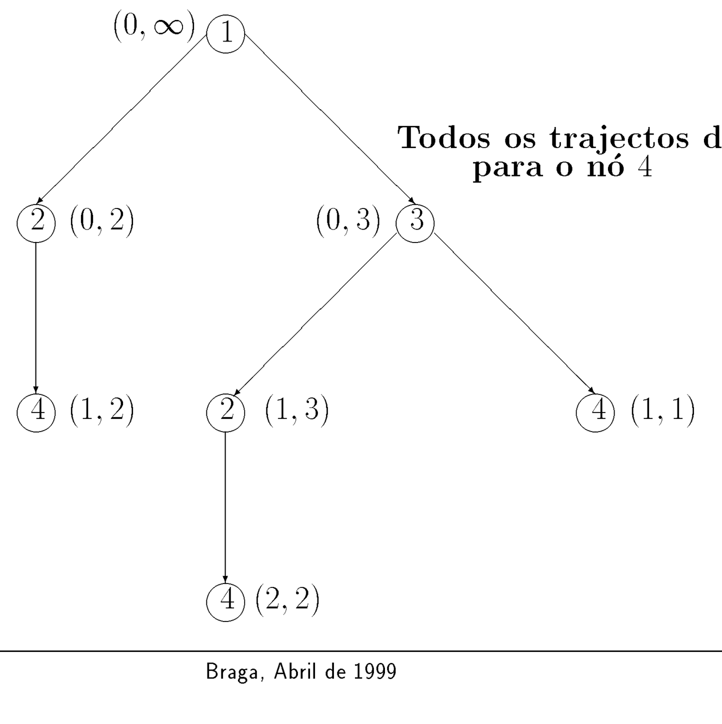
ALGORITMO DE ROTULAÇÃO PARA O PROBLEMA 1

** $\pi_i \longleftarrow$ distância actual do trajecto de s para i **
** $\theta_i \longleftarrow$ capacidade actual do trajecto de s para i **

$(\pi_i, \theta_i) \longleftarrow (\infty, 0)$, $\forall\ i \in \mathcal{N}$
$(\pi_s, \theta_s) \longleftarrow (0, \infty)$
$X \longleftarrow \{s\}$

Enquanto $X \neq \emptyset$ **faz**:
    $i \longleftarrow$ nó de X
    $X \longleftarrow X - \{i\}$
    Para todo $j \in \mathcal{N}$ **tal que** $(i, j) \in \mathcal{A}$ **faz:**
        Se $\pi_i + c_{ij} < \pi_j$ **então:**
            $\pi_j \longleftarrow \pi_i + c_{ij}$
            $\theta_j \longleftarrow \min\ \{\theta_i, u_{ij}\}$
            Se $j \notin X$ **então** $X \longleftarrow X \cup \{j\}$
        senão
            Se $\pi_i + c_{ij} = \pi_j$ **e** $(\min\{\theta_i, u_{ij}\} > \theta_j)$ **então**
                $\theta_j \longleftarrow \min\{\theta_i, u_{ij}\}$
                Se $j \notin X$ **então** $X \longleftarrow X \cup \{j\}$





Todos os trajectos do nó 1 **para o nó** 4

PORQUÊ ?

NEM TODOS os trajectos de capacidade máxima são constituídos por subtrajectos de capacidade máxima.
Assim, os algoritmos de rotulação apenas consideram um subconjunto de trajectos de capacidade máxima.

COMO ?

TEOREMA:

Seja $\varepsilon = \min \{u_{ij} \ : \ (i,j) \in \mathcal{A}\}$.
Então $u(p) \geq \varepsilon, \ \forall \ p \in \mathcal{P}$.

ALGORITMO:

$u^* \longleftarrow \max \{u(p) \ : \ p \in \mathcal{P}\}$;
$\mathcal{A}^* \longleftarrow \mathcal{A} - \{(i,j) \ : \ u_{ij} < u^*\}$;
Determinar o trajecto mais curto em $(\mathcal{N}, \mathcal{A}^*)$.