

## CAPÍTULO V

### V.1 - ALGORITMOS DE COMPLEXIDADE POLINOMIAL (Introdução):

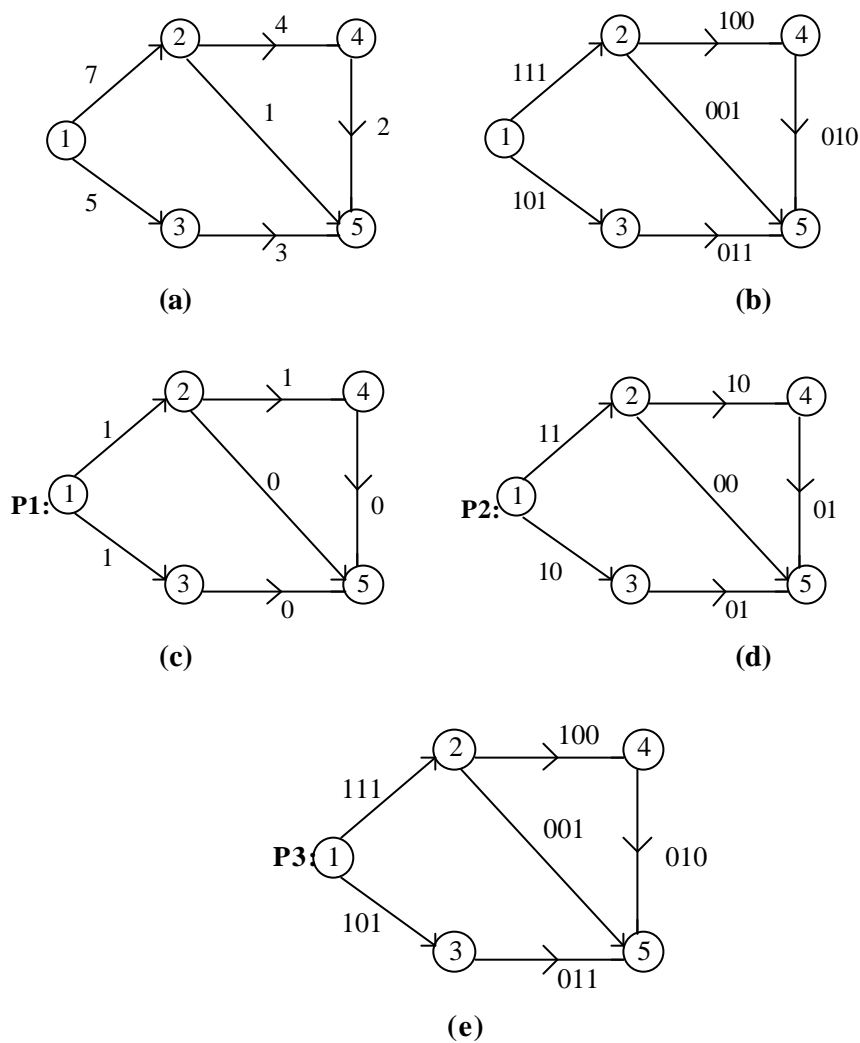
No desenvolvimento de algoritmos polinomiais de fluxo em rede quatro abordagens principais são utilizadas: incremento geométrico, escala de capacidade (ou custo), programação dinâmica e busca binária. Fazemos aqui uma abordagem sobre a escala de capacidade, daremos inicialmente uma descrição do método através de um exemplo (*bit scaling*), e posteriormente apresentamos o algoritmo de caminhos mínimos sucessivos utilizando a escala de capacidade.

A utilização da escala é uma técnica poderosa em vários problemas combinatórios. Os algoritmos assim implementados, começam satisfazendo condições de otimalidade de forma relaxada, obtendo, dessa forma, uma solução inicial aproximada. Se as condições de otimalidade são violadas  $\Delta$  unidades inicialmente (onde  $\Delta$  pode ser consideravelmente grande: C ou U por exemplo), no passo seguinte reduzimos  $\Delta$  à metade e reotimizamos o problema. Reduzimos novamente  $\Delta$  pela metade e fazemos uma nova reotimização até que  $\Delta$  seja tão pequeno quanto se queira. Esta estratégia é bastante flexível e permite a construção de algoritmos distintos dependendo das condições de otimalidade a serem relaxadas e da forma de se fazer as reotimizações.

Trabalharemos com uma versão modificada do algoritmo de caminhos mínimos sucessivos. Na fase  $\Delta$ -escala permitimos que as restrições de oferta/demanda sejam violadas  $\Delta$  unidades de fluxo e que a rede residual contenha ciclos de custo negativo. O algoritmo resultante reduz o número de cálculos na obtenção do caminho mínimo de  $nU$  para  $m \log U$ .

### V.2 - ESCALA DE CAPACIDADE: (Descrição do Método)

Descrevemos inicialmente a forma mais simples de escala, chamada *bit-scaling*. Neste método representamos os dados como números binários e resolvemos um problema P parametricamente como uma sequência de problemas  $P_1, P_2, \dots, P_k$ . O problema  $P_1$  toma sempre o primeiro bit (mais significativo), o problema  $P_2$  toma os dois primeiros bits (mais significativos) e assim sucessivamente até obtermos  $P_k = P$ . Se considerarmos um problema de fluxo em rede onde U é o valor do arco de maior capacidade, teremos  $k = \lceil \log U \rceil$  "será" o número de bits para se representar U. Representaremos também os demais arcos com k-bits adicionando-se zeros à esquerda de cada capacidade. Desta forma o problema  $P_k$  considera problemas cuja capacidade de cada arco é representada pelos k-bits mais significativos. Abaixo representamos um exemplo deste tipo de escala:



**Figura V.1: (a) Rede com Capacidades; (b) Representação em Binário;  
(c)-(e) Problemas P1,P2 e P3**

Para  $k=2,3,\dots,K$ , a solução ótima do problema  $P_{k-1}$  serve como ponto de partida para a solução de  $P_k$ . A técnica de escala é útil sempre que uma reotimização a partir de subproblemas mais fáceis é mais eficiente que a resolução do problema original diretamente.

**Importante!** Observe que a capacidade de um arco no subproblema  $P_k$  é o dobro da capacidade do arco correspondente em  $P_{k-1}$  mais 0 ou 1. •

O algoritmo abaixo descreve genericamente a idéia presente na técnica bit-scaling:

**Algoritmo:** *Bit-Scaling*;

**Início**

Obter uma solução ótima de  $P_1$ ;

**Para  $k:=2$  até  $K$  faça**

Reotimize  $P_k$  utilizando uma solução ótima de  $P_{k-1}$ ;

**fim.**

Note que o número de reotimizações é de ordem  $O(\log U)$  passos (número de bits do maior arco). Desta forma, a reotimização precisa ser apenas um pouco mais eficiente que a resolução do problema original diretamente (otimização).

**Exemplo V.1:** Vejamos, por exemplo, o problema de fluxo máximo. Seja  $v_k$  o valor do fluxo máximo para o problema  $P_k$  e seja  $x_k$  o fluxo máximo correspondente a  $v_k$ . No problema  $P_k$ , a capacidade de cada arco será duas vezes sua capacidade em  $P_{k-1}$  mais 0 ou 1. Se multiplicamos o fluxo máximo  $x_{k-1}$  de  $P_{k-1}$  por 2, obtemos um fluxo viável para  $P_k$  (verificação trivial). Observe que  $v_k \leq 2v_{k-1} + m$ , pois multiplicando-se por 2 o fluxo de cada arco de  $x_{k-1}$  e somando 1 unidade teremos uma cota superior para o fluxo máximo de  $P_k$ , assim  $v_k - 2v_{k-1} \leq m$  (a variação entre 2 fluxos consecutivos será no máximo  $m$ ).

Em geral é mais fácil a reotimização do problema de fluxo máximo do que a resolução do problema diretamente. Por exemplo, o algoritmo clássico de rotulação (Ahuja, seção 6.5) realiza a reotimização em no máximo  $m$  acréscimos de fluxo requerendo um total de  $O(m^2)$  operações. Desta forma o algoritmo bit-scaling executará um total de  $O(m^2 \log U)$  operações, o que significa uma grande melhora na complexidade já que o algoritmo clássico de rotulação tem complexidade total de  $O(mnU)$  operações. •

**V.3) CAMINHOS MÍNIMOS SUCESSIVOS COM ESCALA DE CAPACIDADE:**

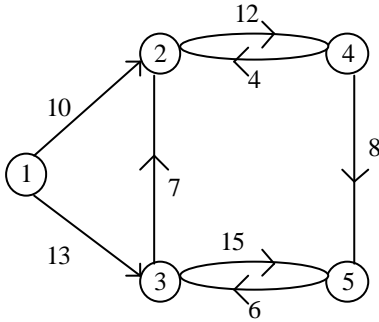
Uma abordagem alternativa da utilização da escala é considerar uma sequência de problemas  $P(1), P(2), \dots, P(k)$  cada um manipulando os dados originais do problema. Neste caso entretanto, resolvemos cada problema  $P(k)$  aproximadamente com erro  $\Delta_k$ . Fazemos inicialmente  $\Delta_1$  suficientemente grande e então reduzimos sucessivamente este valor.

Podemos imaginar que se  $x_k$  é uma solução ótima associada ao problema  $P_k$  (erro  $\Delta_k$ ), modificando-se os dados (capacidades) em no máximo  $\Delta_k$  unidades e resolvendo este novo problema, teremos chegado a uma nova solução ótima. Em outras palavras, começamos com as condições de otimalidade (p/ o problema de fluxo mínimo), mas, ao invés de obter estas condições

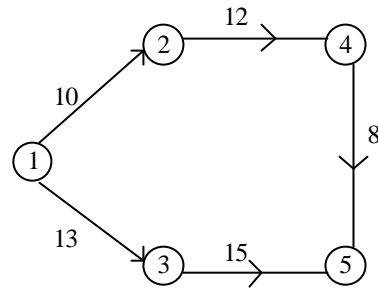
de forma exata, geramos um problema "aproximado" ao original onde as condições de otimalidade devem ser satisfeitas. Inicialmente escolhemos  $\Delta$  suficientemente grande (por exemplo  $U$ ) e facilmente encontramos uma solução inicial que satisfaz as condições de otimalidade para o problema relaxado. Trocamos o parâmetro  $\Delta$  por  $\Delta/2$  e reotimizamos o novo problema utilizando a solução anterior. Repetimos o processo reduzindo-se  $\Delta$  até obtermos  $\Delta \leq 1$  (como será visto adiante). Abaixo definimos a rede residual associada ao parâmetro  $\Delta$ .

**Definição V.1:** Dados um fluxo  $x$  e um parâmetro  $\Delta$  definimos *rede  $\Delta$ -residual* como uma rede que contém arcos com capacidade residual pelo menos  $\Delta$ .

**Exemplo:** Abaixo observamos uma rede onde são representados apenas as capacidades ao lado de cada arco.



(a) Rede Residual  $G(x)$



(b) Rede  $\Delta$ -Residual  $G(x, \Delta)$  p/  $\Delta=8$

Na rede 8-residual as capacidades de cada arco são de pelo menos 8 unidades. Note que se  $\Delta=1$  teremos  $G(x, \Delta) = G(x)$  ( $G(x, \Delta)$  é um subgrafo de  $G(x)$ ). Os custos associados a cada arco permanecem inalterados em  $G(x, \Delta)$ . ♦

O algoritmo com escala de capacidade é uma variante do algoritmo de caminhos mínimos sucessivos. Utilizamos um pseudofluxo  $x$  e desajustes  $e(i)$  como definido no capítulo anterior. Dado um parâmetro  $\Delta$  inicial o algoritmo mantém um pseudofluxo satisfazendo as condições de otimalidade de custo reduzido em  $G(x, \Delta)$  e gradualmente converte este pseudofluxo em fluxo identificando caminhos mais curtos de nós com excesso (neste caso  $e(i) \geq \Delta$ ) a nós com déficit ( $e(i) \leq -\Delta$ ) e aumentando o fluxo nestes caminhos. Denominaremos esta fase de fase  $\Delta$ -escala ( $\Delta$  fixo). Inicialmente fazemos  $\Delta = 2^{\lfloor \log U \rfloor}$ . O algoritmo faz com que em cada fase  $\Delta$ -escala a variação de fluxo em cada caminho aumentante seja de  $\Delta$  unidades de fluxo. Quando não for possível (nenhum nó tem excesso  $e(i) \geq \Delta$  ou déficit  $e(i) \leq -\Delta$ ) trocamos  $\Delta$  por  $\Delta/2$  e continuamos

o processo até obtermos  $\Delta=1$ . Resolvendo-se  $G(x, \Delta)$  para  $\Delta = 1$  teremos chegado ao fluxo ótimo. Note que teremos satisfeito as condições de otimalidade de custo reduzido.

Dado um parâmetro  $\Delta$  definimos  $S(\Delta)$  e  $T(\Delta)$  como:

$$S(\Delta) = \{i : e(i) \geq \Delta\}$$

$$T(\Delta) = \{i : e(i) \leq -\Delta\}$$

Na fase  $\Delta$ -escala, cada aumento de fluxo deve começar em um nó de  $S(\Delta)$  e finalizar em um nó de  $T(\Delta)$  (caminho mais curto). Note ainda que cada aumento de fluxo ocorre em arcos com capacidade pelo menos  $\Delta$ .

Cada arco de  $G(x, \Delta)$  satisfaz as condições de otimalidade de custo reduzido, entretanto, os arcos que estão em  $G(x)$  e não estão em  $G(x, \Delta)$  podem violar estas condições de otimalidade.

Abaixo apresentamos o algoritmo de caminhos mínimos sucessivos com escala de capacidade. Os valores  $r_{ij}$  representam as capacidades dos arcos na rede residual. Como descrito abaixo, caso tenhamos  $r_{ij} \geq \Delta$  e  $\bar{c}_{ij} < 0$  enviamos  $r_{ij}$  unidades de fluxo no arco  $(i,j)$ , satisfazendo assim as condições de otimalidade de custo reduzido na rede  $\Delta$ -residual (ou seja, devemos ter  $\bar{c}_{ij} \geq 0 \quad \forall (i,j) \in G(x,\Delta)$ ).

#### **Algoritmo:** Escala de Capacidade

##### **Início**

$x := 0$ ;

$\Delta = 2^{\lfloor \log U \rfloor}$ ;

**Enquanto**  $\Delta \geq 1$  **faça**

##### **Início**

**Para** todo arco  $(i,j)$  na rede residual  $G(x)$  **faça**

**Se**  $r_{ij} \geq \Delta$  e  $\bar{c}_{ij} < 0$  **então** envie  $r_{ij}$  unidades de fluxo no arco  $(i,j)$ ;

Atualize  $x$  e  $e(i)$ ;

**fim**;

$S(\Delta) = \{i : e(i) \geq \Delta\}$ ;

$T(\Delta) = \{i : e(i) \leq -\Delta\}$ ;

**Enquanto**  $S(\Delta) \neq \emptyset$  e  $T(\Delta) \neq \emptyset$  **faça**

##### **Início**

Selecione  $k \in S(\Delta)$  e  $l \in T(\Delta)$ ;

Determine os rótulos  $d(\cdot)$  do nó  $k$  à todos os outros nós em  $G(x,\Delta)$  com respeito a  $\bar{c}_{ij}$ ;

Seja  $P$  o caminho mais curto do nó  $k$  ao nó  $l$  em  $G(x, \Delta)$ ;

Atualize  $\pi := \pi - d$ ;  
 Aumente  $\Delta$  unidades de fluxo no caminho  $P$ ;  
 Atualize  $x$ ,  $S(\Delta)$ ,  $T(\Delta)$  e  $G(x, \Delta)$ ;  
**fim**;  
 $\Delta := \Delta/2$ ;  
**fim**;  
**fim**.

Como observado acima na fase  $\Delta$ -escala, o algoritmo primeiro verifica se todos os arcos  $(i, j)$  na rede  $\Delta$ -residual satisfazem as condições de otimalidade de custo reduzido, ou seja,  $\bar{c}_{ij} \geq 0, \forall (i, j) \in G(x, \Delta)$ . Os arcos  $(i, j)$  da rede  $\Delta$ -residual que não satisfazem as condições de otimalidade de custo reduzido ( $\bar{c}_{ij} < 0$ ) são imediatamente saturados e eliminados (pois  $r_{ij} = u_{ij} - x_{ij} = 0$ ). O arco contrário  $(j, i)$  possui custo  $\bar{c}_{ji} = -c_{ij} > 0$ .

Na fase  $\Delta$ -escala, cada incremento de fluxo começa em um nó  $k \in S(\Delta)$  e termina em um nó  $l \in T(\Delta)$ , transportando  $\Delta$  unidades de fluxo. Supondo que  $G(x, \Delta)$  contém um caminho direcionado sem restrições de capacidade entre cada par de nós (hipóteses adicionais, capítulo III item iii) sempre teremos um caminho mínimo entre  $k$  e  $l$ .

### V.3.1 - Análise de Complexidade:

Comparativamente aos algoritmos de caminhos mínimos sucessivos (visto no capítulo anterior) este algoritmo de escala de capacidade permite que uma quantidade maior de fluxo seja enviada a cada fase  $\Delta$ -escala diminuindo consideravelmente o número total de iterações no pior caso.

Este método permite uma melhora na complexidade de  $O(nUS(n, m, nC))$  para  $O(m \log U \cdot S(n, m, C))$  (vide Ahuja) onde  $U$  é o valor do arco de maior capacidade e  $S(n, m, C)$  é a complexidade do problema de caminho mínimo da rede com  $n$  nós e  $m$  arcos sendo todos os custos não negativos e limitados por  $C$  (observe, analogamente ao capítulo anterior que na avaliação da complexidade  $S(n, m, nC)$  é utilizada no lugar de  $S(n, m, C)$  pois os custo reduzidos são limitados por  $nC$ )