ESTUDO COMPARATIVO DE ALGORITMOS DE FLUXO MÁXIMO

Angel Guillermo Coca-Balta

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

Aprovada por:

Antonio A F Oliveira
Antônio Alberto Fernandes Oliveira
(Orientador)

Clóvis Caesar Gonzaga

Jayme Luiz Szwarcfiter

RIO DE JANEIRO, RJ - BRASIL

OUTUBRO DE 1983

COCA-BALTA, ANGEL GUILLERMO

. Estudo Comparativo de Algoritmos de Fluxo Máximo. |Rio de Janeiro| 1983.

x , 96 p. 29,7 cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1983.

Tese - Universidade Federal do Rio de Janeiro, COPPE.

1. Otimização I.COPPE/UFRJ II.Título(Série).

DEDICATÓRIA

À Judith, um belo exemplo de esposa e mãe, e a Cecília e Guillermo, os nossos filhos, pelo seu sacrifício e dias de solidão, me incentivando no desenvolvimento deste trabalho.

AGRADECIMENTOS

- Ao professor Antônio Alberto F. de Oliveira, pela sua orientação e encorajamento no desenvolvimento deste trabalho, dedicando-se como um grande Mestre e Amigo.
- Aos colegas e amigos Manuel B.Lino Salvador e Manuel A. Milla Miranda, pelo apoio moral e acadêmico.
- Ao CNPq no Brasil, pelo apoio financeiro.
- À "Universidad Nacional Mayor de San Marcos", no Perú, pela licença concedida para fazer o curso de Pós-Graduação.
- Ao professor Flávio Vega Villanueva, pelo apoio que sempre deu aos colegas da U.N.M.S.M. visando elevar o nível acadêmico pessoal e institucional. Em particular, pelo seu constante apoio moral a mim dedicado.
- Aos meus pais pelo incentivo que sempre deram aos meus estudos, desde a Escola Primária até a Pós-Graduação.
- A todos aqueles que de alguma forma contribuiram para realização deste trabalho.

RESUMO

Um dos temas tratados em Programação Combinatória, com variados enfoques e que, sucessivamente, visam melhorar os outros, é o chamado Problema de Fluxo Máximo. Neste trabalho descrevemos e analisamos as principais abordagens para o tratamento desse problema e que não se constituem em mera sofisticação da estrutura de dados utilizada, como acontece com uma série de métodos recentes.

Essas abordagens principais, conhecidas pelos nomes de seus autores, são as de Ford e Fulkerson, Edmonds e Karp, Dinic, Karzanov e de Malhotra-Kumar-Maheshwari.

ABSTRACT

A Combinatorial Programming topic with several different kinds of methods for its solution is the Max Flow Problem. This work describes and analyses the main methods for the treatment of that problem. These methods are simple and without a sophisticated data structure as some new methods have.

That main methods are named with the name of their authours: Ford and Fulkerson, Edmonds and Karp, Dinic, Karzanov and Malhotra-Kumar-Maheshwari.

RESUMEN

Uno de los temas tratados en Programación Combinatória, con variados enfoques, los cuales, sucesivamente mejoran a los otros, el el llamado Problema de Flujo Máximo. En este trabajo describimos y analisamos los principales métodos para el tratamiento de aquel problema y que no se constituyen simples sofisticaciones de la estructura de datos utilizada, conforme ocure con métodos recientes.

Esos métodos principales, conocidos por los nombres de sus autores son los de Ford y Fulkerson, Edmonds y Karp, Dinic, Karzanov y el de Malhotra-Kumar-Maheshwari.

ÍNDICE

CAPÍTULO 1

O PROBLEMA DE FLUXO MÁXIMO: ABORDAGENS CLÁSSICAS

1.1. INTRODUÇÃO

Um dos problemas de Programação Combinatória que tem recebido um grande número de enfoques para obter a sua solução é o Problema de Fluxo Máximo. Primeiro foi a abordagem de Ford e Fulkerson |[10]| em 1956 e a partir de 1969, múltiplas pesquisas são feitas na U.S.A, na Rússia e Israel, sendo que algumas delas não tiveram a divulgação necessária por motivos de tradução.

O Problema de Fluxo Máximo é basicamente um Problema de Programação Linear e portanto pode ser resolvido pelos métodos próprios desses tipo de problema, mas dadas as características particulares dele pode-se obter algoritmos que o afastam dos métodos de solução daqueles conhecidos problemas de Programação Linear.

1.2. DEFINIÇÕES PRÉVIAS

1.2.1. Grafo Orientado

Um Grafo Orientado G = (V,E) é uma estrutura formada por um conjunto finito não vazio V de elementos, chamados de Nós ou vértices, e por o conjunto E de pares ordenados de elementos de V e que chamaremos de Arcos ou Ramos.

Notação: Dado o grafo $G = (V,E)$

Se $u \in V$, $v \in V \Rightarrow e = (u,v) \in E$ ou

$\bar{e} = (v,u) \in E$ (*)

Os conjuntos V e E são finitos. Os cardinais serão $|V| = n$, $|E| = m$.

1.2.2. Arcos Diretos e Contrários

Se u e v são dois nós num grafo, chamaremos de Arco Direto a um dos dois pares, o outro par será chamado de Arco Contrário.

$e = (u,v)$ ← Arco Direto

$\bar{e} = (v,u)$ ← Arco Contrário

1.2.3. Nós Sucessores e Antecessores

Se v é um nó num grafo $G = (V,E)$ chamaremos de Nó sucessor de v, a todo nó x, tal que $e = (v,x)$ é um arco do grafo. Se denotamos com $S(v)$ ao conjunto de nós sucessores de v, temos:

$$S(v) = \{x \in V \ / \ (v,x) \in E\}$$

(*) Quando for necessário fazer diferença do arco "e" com outro símbolo, escreveremos: "$\underline{e}$".

Similarmente se A(v) denota os nós antecessores de v, temos:

$$A(v) = \{x \in V \ / \ (x,v) \in E\}$$

1.2.4. Arcos Sucessores e Antecessores

Se v é um nó num grafo G = (V,E) e se x é um nó sucessor de v, chamaremos de Arco Sucessor de v à todo arco da forma e = (v,x). Se denotamos como $\overline{S}(v)$ ao conjunto de arcos sucessores de v, temos:

$$\overline{S}(v) = \{(v,x) \ / \ x \in S(v)\}$$

Similarmente se $\overline{A}(v)$ denota os arcos antecessores de v, temos:

$$\overline{A}(v) = \{(x,v) \ / \ x \in A(v)\}$$

1.2.5. Capacidade de um Arco

Em um grafo G = (V,E), estabeleceremos a função c de E em $\mathbb{R}_0^+$ (1) que atribue a cada arco um número real positivo.

$$c: \quad E \longrightarrow \mathbb{R}_0^+$$
$$e = (u,v) \longrightarrow c(e) \geq 0$$

A função c é chamada de Capacidade do arco.

---

(1) $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$

1.2.6. Rede Capacitada

Se no grafo G = (V,E) estabelecemos a função Capacidade do Arco, teremos então a estrutura chamada de Rede Capacitada G = (V,E,c).

Para os fins deste trabalho serão considerados dois nós especiais chamados de Fonte (s) e Poço (t). Temos então a notação a seguir:

Rede capacitada: G = (V,E,c)
Fonte: s sendo que $s \in V$
Poço: t sendo que $t \in V$
Vértices: V
Ramos E
V-{s,t}: $\overline{V}$

1.2.7. Fluxo na Rede (Da Fonte até o Poço)

Se G = (V,E,c) é uma rede, chama-se de Fluxo da fonte até o poço a função real f de domínio igual ao conjunto E de arcos tal que:

$$\sum_{u \in S(v)} f(v,u) - \sum_{w \in A(v)} f(w,v) = \begin{cases} F & \text{se } v = s \\ 0 & \text{se } v \in \overline{V} \\ -F & \text{se } v = t \end{cases} \qquad (1)$$

$$0 \leq f(u,v) \leq c(u,v), \quad e = (u,v) \in E \qquad (2)$$

Quando for preciso, utilizaremos a notação G = (V,E,c,f) para indicar que um fluxo f é associado à rede G = (V,E,c).

Se procuramos uma interpretação do conceito apresentado anteriormente, é fácil entender que da fonte s sai um determinado "fluxo" de valor F, o qual chega até o poço t, e que nos nós intermediários o "fluxo" é conservado no sentido de que cada nó serve somente para fazer uma conexão entre os arcos antecessores e sucessores. A utilidade de cada arco e = (u,v) é então de facilitar o passo do fluxo desde o nó u até o nó v, logicamente com a restrição assinalada pela capacidade.

Na procura da solução do problema que descreveremos na próxima seção utilizaremos os arcos para acrescentar o fluxo existente numa rede, neste sentido se u e v são nós e se e = (u,v) é um arco direto, por ele pode-se levar um acréscimo do fluxo desde o nó u até o nó v, no entanto acharemos casos em que estando situados no nó u tenhamos que ir até o nó w, sendo que $\underline{e}$ = (w,u) é um arco direto com um determinado fluxo f(e), não nulo, neste caso faremos uso do arco $\overline{e} = (u,w) \notin E$ com capacidade f(e) o que na realidade é uma eliminação total ou parcial do fluxo f(e) já existente. Segundo a exposição que faremos após, um arco $\underline{e}$ serve então para acrescentar o fluxo ou para diminuí-lo se for necessário.

1.2.8. Capacidade Residual Relativa ao Fluxo f

Dados dois nós adjacentes numa rede capacitada G = (V,E,c) e um fluxo f, definimos Capacidade Residual de e = (u,v) em relação ao fluxo f, e denotamos $\delta_f(e)$, da seguinte forma:

$$
\begin{aligned}
&\text{- se} \quad e = (u,v) \in E \quad \Rightarrow \quad \delta_f(e) = c(e) - f(e) \geq 0 \\
&\text{- se} \quad e = (u,v) \notin E, \quad \bar{e} = (v,u) \in E \quad \Rightarrow \quad \delta_f(e) = f(\bar{e}) > 0
\end{aligned}
\tag{3}
$$

Na figura 1.1 apresentamos uma rede G = (V,E,c), onde as capacidades são expressas dentro de colchetes, e o fluxo é expresso sem aquele símbolo.

G = (V,E,c,f)

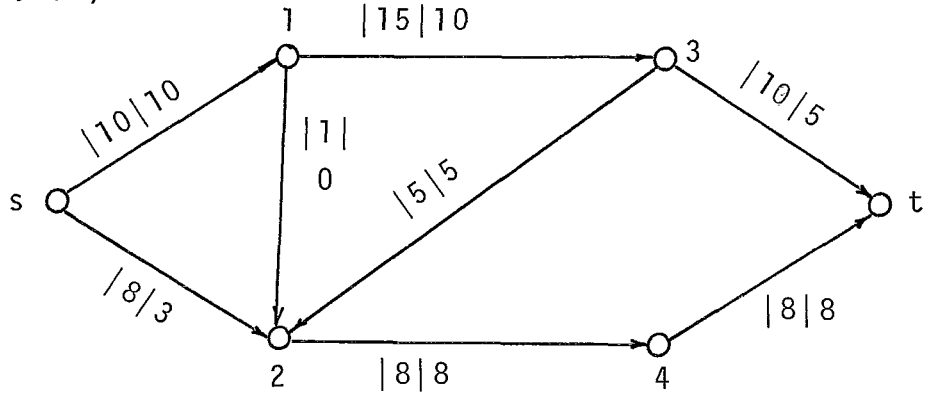


Figura 1.1

$\delta(s,1) = 10 - 10 = 0$	$\delta(1,s) = 10$
$\delta(1,3) = 15 - 10 = 5$	$\delta(3,1) = 10$
$\delta(1,2) = 1 - 0 = 1$	$\delta(2,1) = 0$
$\delta(s,2) = 8 - 3 = 5$	$\delta(2,s) = 3$
$\delta(3,2) = 5 - 5 = 0$	$\delta(2,3) = 5$
$\delta(2,4) = 8 - 8 = 0$	$\delta(4,2) = 8$
$\delta(4,t) = 8 - 8 = 0$	$\delta(t,4) = 8$
$\delta(3,t) = 10 - 5 = 5$	$\delta(t,3) = 5$

G'(V,E',c',f)

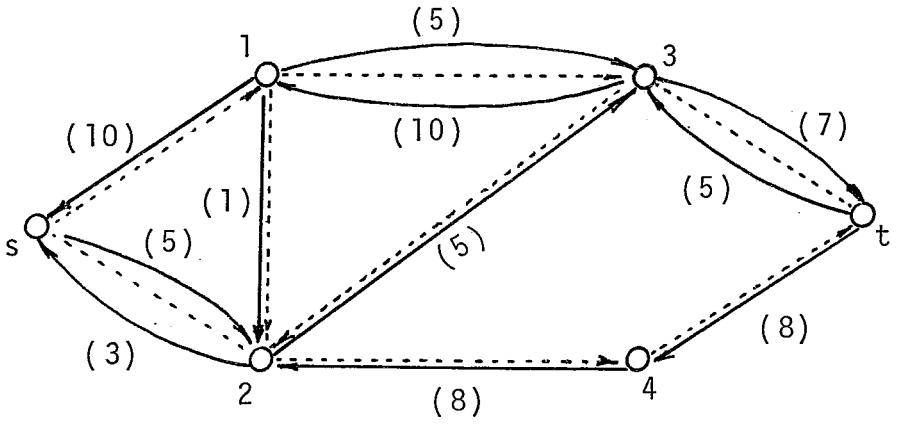


Figura 1.2

1.2.9. Rede Residual Relativa ao Fluxo f

Seja G = (V,E,c) uma rede capacitada e seja f um fluxo na rede, chamamos de Rede Residual relativa ao fluxo f a rede capacitada $G'_f = (V, E'_f, c')$, (*), onde dados os nós $u \in V$, $v \in V$, o arco $e = (u,v)$ pertence ao conjunto E'_f se somente se

1) u e v são adjacentes em G

2) $\delta_f(e) > 0$

(*) Para não sobrecarregar a notação, quando o fluxo f estiver perfeitamente definido escreveremos simplesmente G', E', ...

Fazemos então para todo arco e de E'_f

$$c'(e) = \delta_f(e)$$

Na figura 1.2 apresentamos a Rede Residual G' = (V,E',c') correspondente a rede da figura 1.1, onde a linha pontilhada é a correspondente rede original G.

1.2.10. Arco Útil

Se f é um fluxo na rede G = (V,E,c), chama-se de Arco Útil a todo arco da correspondente rede residual G'.

1.2.11. Arco Saturado

Chama-se de Arco Saturado, a todo arco da rede G com fluxo igual a sua capacidade.

Se observarmos as figuras 1.1 e 1.2, e segundo as definições 1.2.8 até 1.2.11, temos:

- Todo arco saturado da rede G, origina na rede residual G' um arco útil de sentido contrário.
- Todo arco $e \in E$ não saturado e de fluxo $f(x) > 0$, origina dois arcos úteis, o direto (o próprio) e o contrário na rede residual G'.
- Todo arco e = (u,v) não saturado e com fluxo nulo na rede G, é o próprio na rede G'.

Segundo o comentário feito logo depois da definição 1.2.7 e pelas definições 1.2.8 e 1.2.9, cada arco e = (x,y) na rede residual G' = (V,E',c') permite levar um fluxo f_a, com $f_a(e) \leq c'$, desde o nó x até o nó y, acrescentando o fluxo já existente na rede G. Este escoamento de fluxo se reflete na rede original G = (V,E,c) da seguinte maneira:

- se e = (x,y) $\in$ E, então o fluxo existente é acrescentado para a quantidade de fluxo que levamos do nó x até o nó y.
- se e = (x,y) $\notin$ E, então $\bar{e}$ = (y,x) $\in$ E e o fluxo existente é diminuido para a quantidade de fluxo que levamos do nó x até o nó y.

1.2.12. Caminho de Acréscimo de Fluxo entre s e t para o Fluxo f. (c.a.f.)

Dada uma rede capacitada G e um fluxo f indo da fonte s ao poço t chama-se de Caminho Acréscimo de Fluxo para f a um caminho L de s a t em G'_f

Segundo o comentário feito na definição acima, fica claro o significado da palavra "acréscimo".

1.2.13. Comprimento do Caminho

Chama-se Comprimento do Caminho L ao número de seus arcos.

1.2.14. Distância Entre Dois Nós

Distância d(u,v) entre dois nós u e v é o comprimento do caminho mais curto entre eles.

1.2.15. Caminho f-Bloqueado

Se L é um caminho numa rede capacitada G = (V,E,c) e f um fluxo na rede, chamamos de Caminho f-Bloqueado ou simplesmente Bloqueado se existir um arco saturado no caminho.

1.2.16. Arco f-Bloqueado

Um arco e = (u,v) numa rede G = (V,E,c,f), é chamado de Arco f-Bloqueado se qualquer caminho da forma (u,v,w,..,t) é f-Bloqueado.

1.2.17. Nó f-Bloqueado

Um nó u é f-Bloqueado se qualquer caminho de u a t é f-Bloqueado.

1.2.18. Fluxo Maximal

Dada uma rede capacitada G, diz-se que um fluxo é Maximal se todo caminho L em G ligando a fonte com o poço contém um arco saturado por f.

Em outras palavras podemos afirmar que f é fluxo maximal numa rede G se a fonte s é f-bloqueada.

Observamos que se trocarmos na definição acima "...todo caminho L em G ..." por "... todo caminho L em G'_f ..." então f será de fato o fluxo máximo que pode ir da fonte ao poço na rede G. Esse fato será demonstrado posteriormente.

1.3. O PROBLEMA DE FLUXO MÁXIMO

Seja G = (V,E,c) uma rede capacitada com valores inteiros, seja também f um fluxo da fonte s até o poço t. O Problema de Fluxo Máximo é o de achar aquele fluxo f de maior valor.

Segundo a definição 1.2.7, o Problema é descrito a seguir:

Maximizar: F

Sujeito a:

$$\sum f(u,v) - \sum f(w,u) = \begin{cases} F & \text{se } u = s \\ 0 & \text{se } u \in \overline{V} \\ -F & \text{se } u = t \end{cases}$$

$$0 \leq f(u,v) \leq c(u,v)$$

Uma primeira abordagem para a solução do problema foi feita por Ford e Fulkersen [10] em 1956, apresentando o Teorema do Corte Mínimo - Fluxo Máximo que descreveremos posteriormente. Ante vejamos:

1.3.1. Corte Separando s e t (Corte s-t)

Seja G = (V,E,c) uma rede capacitada, e seja X um subconjunto de nós incluindo a fonte s, seja também $\overline{X} = V - X$ o complementar correspondente e que incluo o poço t, chama-se de Corte Separado s e t ou simplesmente Corte s-t e denotamos com $(X,\overline{X})$, ao conjunto de arcos e = (u,v) da rede G, onde u é um nó em X e v é um nó em $\overline{X}$.

Seja por exemplo a rede da figura 1.1, se X = {s,1,3}, o complementar $\overline{X}$ = {2,4,t}, então o Corte s-t é

$$(X,\overline{X}) = \{(1,2), (3,2), (s,2), (3,t)\}$$

Vejamos agora, seja L um caminho qualquer que liga s a t, é fácil compreender que pelo menos um dos arcos dele pertence ao corte, fato que podemos usar para afirmar que se na rede G, tiramos os arcos de um corte, então não existirá caminho ligando s a t.

1.3.2. Capacidade do Corte

Seja G = (V,E,c) uma rede capacitada e seja X um subconjunto dos nós em V que contém a fonte s. Seja também o corte $(X,\overline{X})$, chama-se de Capacidade do Corte $c(X,\overline{X})$ ao somatório das capacidades dos arcos que pertencem ao corte, ou seja:

$$c(X,\overline{X}) = \sum_{(u,v) \in (X,\overline{X})} c(u,v) \qquad (4)$$

Com a finalidade de abordar o Teorema do Corte Mínimo - Fluxo Máximo, apresentamos logo algumas propriedades importantes referentes ao conceito de corte.

LEMA 1 - Dualidade Fraca

O valor F de qualquer fluxo numa rede, é menor ou igual a capacidade de um corte qualquer $(X,\overline{X})$.

Prova

Dados U e W, subconjuntos de V denotemos por f(U,W) ao somatório $\Sigma f(u,w)$ sendo que u é um nó de U e w é um nó de W, simbolicamente

$$f(U,W) = \sum_{\substack{u \in U \\ w \in W}} f(u,w)$$

Por simplicidade se U ou W é constituído de apenas um elemento q, escrevemos f(q,W), (f(U,q), respectivamente) ao invés de f({q}, W), (f(U, {q}), respectivamente).

Reescrevendo as equações de conservação de fluxo (1) temos então:

$$\begin{aligned} f(s,V) - f(V,s) &= F \\ f(v,V) - f(V,v) &= 0 \qquad \text{onde } v \in \overline{V} \\ f(t,V) - f(V,t) &= -F \end{aligned} \tag{5}$$

Somando as equações acima para todo $v \in X$, e considerando que $s \in X$, e $t \notin X$, temos:

$$F = \sum_{v \in X} [f(v,V) - f(V,v)] \tag{6}$$

$$F = f(X,V) - f(V,X) \tag{7}$$

$$= f(X, X \cup \overline{X}) - f(X \cup \overline{X}, X) \tag{8}$$

Como, se Y_1 e Y_2 (W_1 e W_2, respectivamente) são disjuntos, então

$$f(Y_1 \cup Y_2, W) = f(Y_1, W) + f(Y_2, W)$$

($f(Y, W_1 \cup W_2) = f(Y, W_1) + f(Y, W_2)$, respectivamente), ficamos com:

$$F = f(X,\overline{X}) - f(\overline{X},X), \tag{9}$$

ou seja

$$F = \sum_{\substack{u \in X \\ v \in \overline{X}}} f(u,v) - \sum_{\substack{v \in \overline{X} \\ u \in X}} f(v,u) \tag{10}$$

Como $f(v,w) \geq 0$ e $f(u,v) \leq c(u,v)$, temos:

$$\sum_{(u,v) \in (X,\overline{X})} f(u,v) \leq \sum_{(u,v) \in (X,\overline{X})} c(u,v) \qquad (11)$$

$$\sum_{(v,u) \in (\overline{X},X)} f(v,u) \geq 0 \qquad (12)$$

Assim

$$\sum_{(u,v) \in (X,\overline{X})} f(u,v) - \sum_{(v,u) \in (\overline{X},X)} f(v,u) \leq \sum_{(u,v) \in (X,\overline{X})} c(u,v) \qquad (13)$$

o que por (10) e (4) acima, é o mesmo que

$$F \leq c(X,\overline{X}) \qquad (14)$$

C.Q.D.

O lema 1 nos permite afirmar que se tivermos um fluxo f de valor F e um corte $(X,\overline{X})$ tal que $F = c(X,\overline{X})$ então f é um fluxo máximo entre s e t, e $(X,\overline{X})$ é um Corte Mínimo (*) separando s e t (**).

Vamos agora provar que dados uma rede G qualquer e dois de seus nós s e t, existe um fluxo de s a t cujo valor é igual ao de um corte $(X,\overline{X})$ separando s e t.

(*) Corte Mínimo ou de menor capacidade.

(**) Comentário que usaremos na prova do Teorema 1.

TEOREMA 1 - Dados uma rede capacitada G = (V,E,c) e dois nós s e t de V, então as três afirmações abaixo são equivalentes.

1) f é um fluxo máximo entre s e t.
2) Não existe um caminho de acréscimo de fluxo entre s e t para o fluxo f.
3) Existe um corte $(X,\overline{X})$ cuja capacidade é igual ao valor F do fluxo f.

Prova

(1.-) => (2.-)

A prova desta implicação é imediata pelo próprio significado do que é um caminho de acréscimo de fluxo para um fluxo f conforme explicado em 1.2.12 e anteriores.

(2.-) => (3.-)

Considere X o conjunto de nós atingível em G' a partir de s. Se a afirmação (2.-) é verdadeira temos que $t \notin X$ e portanto $(X,\overline{X})$ é um corte separando s e t.

Considere então inicialmente um arco e = (u,v) com $u \in X$ e $v \in \overline{X}$. Se obtém diretamente que

$$f(u,v) = c(u,v) \qquad (15)$$

caso contrário teríamos $v \in X$.

Da mesma forma dado um arco (v,u), com $u \in X$ e $v \in \overline{X}$, temos que

$$f(v,u) = 0 \qquad (16)$$

pois senão $\delta_f(u,v) = f(v,u) > 0$ e teríamos que $v \in X$.

Tomando agora novamente a equação (10)

$$F = \sum_{\substack{u \in X \\ v \in \overline{X}}} f(u,v) - \sum_{\substack{v \in \overline{X} \\ u \in X}} f(v,u)$$

segundo as equações (15) e (16), podemos escrever:

$$F = \sum_{\substack{u \in X \\ v \in \overline{X}}} c(u,v) - 0$$

$$F = c(X,\overline{X})$$

completando a prova dessa implicação.

(3.-) => (1.-)

Esta implicação é uma consequência direta do lema 1 já comentada logo após a prova respectiva.

C.Q.D.

Como um corolário do teorema 1, daremos o seguinte:

TEOREMA 2 - (Teorema "Min Cut-Max Flow") (Dualidade Forte) - A solução do problema de determinação do fluxo máximo entre s e t numa rede capacitada G = (V,E,c) dado em 1.3, é igual a capacidade do corte mínimo em G separando s e t.

Prova

O problema de determinação do fluxo máximo é um Problema de Programação Linear limitado em função do que estabelece o lema 1 e portanto possui uma solução viável onde a função objetivo assume o valor ótimo o que equivale a dizer que existe de fato um fluxo de valor máximo. Dado esse fluxo máximo o teorema 1 nos garante a existência de um corte separando s e t

cuja capacidade é igual ao seu valor e que por conseguinte pelo lema 1 é um corte mínimo separando s e t.

C.Q.D.

O item 2 do teorema 1 motiva a formulação de um primeiro algoritmo para a obtenção de fluxo máximo numa rede, algoritmo em que sucessivamente aumentamos o valor do último fluxo obtido, determinando-se um caminho de acréscimo de fluxo para ele, enquanto isso é possível, e fazendo-se passar por esse caminho um acréscimo $\delta(L)$. Esse algoritmo se deve a Ford e Fulkerson e foi apresentado em 1956 |[2]|, |[10]| e que descreveremos numa próxima seção.

1.4. O TEOREMA MIN.CUT-MAX. FLOW VIA DUALIDADE EM PROGRAMAÇÃO LINEAR

A igualdade entre o valor do fluxo máximo e o do corte mínimo dada pelo Teorema 2 pode ser demonstrada utilizando o Teorema de Dualidade Forte (*) em Programação Linear conforme veremos a seguir.

(*) Esta é a justificativa de termos chamado ao teorema 2 em 1.3 de "Teorema da Dualidade Forte".

Considere então como problema primal o nosso Problema de Fluxo Máximo apresentado na seção 1.3 que reproduzimos a seguir numa forma compacta.

Máx. F

s.a.

$$(I_t - I_s)F + A\underline{f} = \theta \tag{17}$$

$$\underline{f} < \underline{c} \tag{18}$$

$$f \geq \theta \tag{19}$$

Onde: A é a matriz de incidência da rede

$I_t = (0,0,\ldots,0,1)^T$

$I_s = (1,0,\ldots,\ldots,0)^T$

$\underline{f} = (f(s,u),\ldots, f(i,j),\ldots,f(y,t))^T$

$\underline{c} = (c(s,u),\ldots, c(i,j),\ldots,c(y,t))^T$

$(i,j) \in E$

Sejam agora, $h = (h_s,h_u,\ldots,h_i,\ldots,h_t)^T$ a variável dual associada às equações (dos nós) de conservação de fluxo e $w = (w_{sv},w_{sy},\ldots,w_{ij},\ldots,w_{ut})$ a variável dual associada as restrições de capacidade dos arcos.

O Problema Dual (PD) pode então ser expresso por:

$$\text{Min. } Z = \sum_{(i,j) \in E} c(i,j)w_{ij} \tag{20}$$

s.a.

$$h_s - h_t = 1 \tag{21}$$

$$h_j - h_i + w_{ij} \geq 0 \tag{22}$$

$$w_{ij} \geq 0 \tag{23}$$

h_i qualquer

Vamos mostrar agora que o valor ótimo de (PD) é a própria capacidade do corte mínimo e então aplicando o Teorema de Dualidade Forte obteremos a igualdade:

Valor do fluxo máximo = Capacidade do corte mínimo

Para isso observe que o problema (PD) fixadas as variáveis h_i, $i = 1, \ldots |V|$, a melhor escolha para os w_{ij} é dada por

$$w_{ij} = \max.\{0, h_i - h_j\}$$

Fazendo isso podemos reescrever o (PD) agora apenas em função das variáveis h_i da seguinte forma:

$$\min_{\{h/h_s - h_t = 1\}} \left\{ \mathcal{C}(h) = \sum_{(i,j)/h_i > h_j)} c(i,j)(h_i - h_j) \right\} \quad (PD_1)$$

Como $\mathcal{C}(h)$ é função apenas das diferenças $(h_i - h_j)$ é possível acrescentar a todas as suas coordenadas h_i uma constante qualquer sem alterar o seu valor. Isso significa que existe uma solução ótima para o (PD_1) em que $h_t = 0$ e portanto $h_s = 1$. Essas duas igualdades podem então substituir a restrição $h_s - h_t = 1$ sem que o valor ótimo do problema se modifique,

Chame agora esse último problema de (PD_2), e vamos mostrar que sempre existe solução ótima para ele onde

$$h_i \in \{0,1\} \ , \quad i = 1, \ldots, n = |V|.$$

Para isso considere $\overline{h} = (\overline{h}_1 = \overline{h}_s = 1, \overline{h}_2, \ldots, \overline{h}_n = \overline{h}_t = 0)$ uma solução ótima para (PD_2).

Representando por C(h) ao conjunto

$\{k \in \mathbb{R} \ / \ (\exists_i \ / \ h_i = k)\}$,

$\overline{h}$ resolve:

$\min \{|C(h)|, h$ é solução ótima de $(PD_2)\}$

Definimos, então, para todo $k \in \mathbb{R}$

$$H(k) = \{i/h_i = k\}$$

e estabeleça que existe um $\bar{k} \notin \{0,1\}$ tal que $H(\bar{k}) \neq \phi$. Faça ainda $M = \max.\{|h_i|, i = 1,\ldots, n\}$ e considere

$$k^+ = \text{mim } \{2M, \text{mim } \{k > \bar{k}/H(k) \neq \phi\}\}$$

$$k^- = \max \{-2M, \max \{k < \bar{k}/H(k) \neq \phi\}\}$$

Suponha agora que

$$\sum_{h_i > h_j = \bar{k}} c(i,j) > \sum_{\bar{k} = h_i > h_j} c(i,j) \qquad (24)$$

e defina:

$$h' = (h'_1, \ldots, h'_n)$$

onde

$$h'_i = \bar{h}_i \quad \text{se} \quad h_i \neq \bar{k}$$

$$h'_i = k^+ \quad \text{se} \quad h_i = \bar{k}$$

Como $\bar{k} \notin \{0,1\}$ não alteramos o valor de h_s e h_t que continuam valendo 1 e 0 respectivamente. Dessa maneira h' é viável para (PD_2) e além disso teremos

$$\sum_{(i,j)/h'_i > h'_j} c(i,j).(h'_i - h'_j) =$$

$$\sum_{\substack{(i,j)/h_i > h_j \\ h_i, h_j \neq \bar{k}}} c(i,j).(\bar{h}_i - \bar{h}_j) + \sum_{(i,j)/h_i > h_j = \bar{k}} c(i,j).\{(\bar{h}_i - \bar{h}_j) - (k^+ - \bar{k})\} +$$

$$+ \sum_{(i,j)/\bar{k} = h_i > h_j} c(i,j).\{(\bar{h}_i - \bar{h}_j) + (k^+ - \bar{k})\} =$$

$$= \mathcal{C}(\bar{h}) - \{ \sum_{(i,j)/h_i > h_j > \bar{k}} c(i,j) - \sum_{(i,j)/\bar{k} = h_i > h_j} c(i,j) \} (k^+ - \bar{k}) <$$

$$< \mathcal{C}(\bar{h})$$

h' portanto seria uma solução para (PD_2) melhor que $\bar{h}$ em contradição com o fato de que $\bar{h}$ é solução ótima para esse problema.

Da mesma forma se em (24) o sinal de desigualdade for "<" então definindo h" como h' só que trocando k^+ por k^- chegaremos a idêntico resultado.

Assim resta para $\bar{k}$ uma única possibilidade que em (24) tenhamos uma igualdade. Mas nesse caso tomando $\hat{h} = h'$ se $\bar{k} \neq \max \{k/H(k) \neq \phi\}$ e $\hat{h} = h"$, em caso contrário obteremos que $\hat{h}$ será uma solução ótima para (PD_2) e que $|C(\hat{h})| = |C(\bar{h})| - 1$ de novo contradizendo a definição de $\bar{h}$.

Desse modo eliminamos todas as possibilidades de $\bar{k}$ não ser 1 ou 0 e portanto obrigatoriamente $\bar{h}_i \in \{0,1\}$, i = 1,. ,..., n.

Como $\bar{h}$ é solução ótima de (PD_2) podemos então dizer usando também a igualdade dos valores ótimos dos problemas PD, PD_1 e PD_2 que o valor ótimo do problema dual pode ser dado por

$$\min_{h \in \{0,1\}^n / h_s = 1, h_t = 0} \{ \mathcal{C}(h) = \sum_{\substack{(i,j)/h_i = 1 \\ h_j = 0}} c(i,j) \}$$

que finalmente utilizando a bijeção ($h \in \{0,1\}^n \longrightarrow X = \{i/h_i = 1\}$

pode ser escrito

$$\min_{X \ni s} \{ \mathcal{C}(X) = \sum_{\substack{(i,j)/i \in X \\ j \in \overline{X}}} c(i,j) \}$$

que é o próprio problema de determinação da capacidade do corte mínimo e assim verificamos nossa afirmação anterior.

1.5. ALGORITMO DE ROTULAÇÃO DE FORD E FULKERSON (ALGORITMO F-F)

1.5.1. O Algoritmo

Basicamente o algoritmo possui duas rotinas, na primeira ele efetua a busca de um caminho de acréscimo de fluxo e na segunda modifica o fluxo nos arcos desse caminho. Na versão mais usual da primeira rotina, a exploração da rede residual de um fluxo f a partir da fonte é efetuada através de um algoritmo de rotulação em que a cada nó v atingido, associamos um registra de informação (ou rótulo) indicando:

1) O seu antecessor imediato no caminho de s até ele, caminho que o algoritmo utilizou para atingi-lo.

2) Qual o máximo acréscimo de fluxo que podemos escoar por esse caminho.

3) Se o último ramo desse caminho pertence a rede original (se diz num abuso de linguagem, que o ramo é percorrido no sentido direto por esse caminho) ou não (o arco é percorrido no sentido inverso).

Os nós atingidos ou já gerados pelo algoritmo são também chamados de rotulados e de uma forma mais precisa o procedimento efetuado é o seguinte:

Rotina A (ROTU)

Passo 0

Inicialmente todos os nós são não rotulados. Atribuimos então à fonte s o rótulo $(-,\infty)$ e colocamos s na lista dos nós rotulados mas não expandidos, chamada Lista Aberta.

Faça

$s \longleftarrow u$

Passo 1

Se a lista aberta estiver vazia pare, isso indica que não existe caminho de acréscimo de fluxo para f. Caso contrário, escolhemos um nó u ainda não expandido e naturalmente o passamos para a lista dos nós já expandidos (Lista Fechada).

Sejam agora $v_1, v_2, \ldots, v_\ell$ os sucessores de um nó u em G', para cada um deles efetue o seguinte procedimento

Trata-Sucessor

- se v_i já foi rotulado não faça nada, caso contrário,
  - se $(u,v_i) \in E$ e $f(u,v_i) < c(u,v_i)$ crie para v_i o rótulo:

    $(+u, \varepsilon(v_i))$ onde $\varepsilon(v_i) =$

    $= \min \{c(u,v_i) - (f(u,v_i), \varepsilon(u)\}$
  - se $(u,v_i) \notin E$, mas $f(v_i,u) > 0$ atribua à v_i o rótulo

    $(-u, \varepsilon(v_i))$ onde $(v_i) =$

    $= \min \{f(v_i,u), \varepsilon(u)\}$

Em qualquer dos dois casos coloque v_i na lista dos nós não expandidos.

- se v_i = t, vá para a rotina de alteração do fluxo (ROTU)

Fim de Trata-Sucessor.

Quando todos os sucessores já tiverem sido tratados repita o passo 1.

A rotina ROTU é aplicada até atingir o poço t, caso se verifique que o poço não é atingível o algoritmo termina, pois não existindo caminho de acréscimo de fluxo, o fluxo deve ser máximo. Quando o poço é atingido aplica-se a rotina B abaixo, que efetua a atualização do fluxo nos arcos do caminho usado pelo algoritmo para chegar a t. Ela recupera esse caminho determinando os seus nós de trás para a frente com o uso da primeira posição do rótulo.

Rotina B (MUF)

Quando a rotina ROTU foi aplicada e o algoritmo atingiu o poço, ele pode estar rotulado com $(+u, \varepsilon(t))$ ou $(-u, \varepsilon(t))$.

Passo 0

Faça

$u \longleftarrow t$

$\varepsilon \longleftarrow \varepsilon(t)$

Passo 1

- se o nó u é rotulado da forma

$(+v, \varepsilon(u))$, fazer

$f(v,u) \longleftarrow f(v,u) + \varepsilon$, ir para o Passo 2

- se ao contrário o rótulo é

  $(-v, \varepsilon(u))$, fazer

  $f(u,v) \longleftarrow f(u,v) - \varepsilon$, ir para o Passo 2

Passo 2

- se $v \neq s$ fazer

  $u \longleftarrow v$, e voltar para o Passo 1

- se $v = s$

  apagar todos os rótulos e aplicar de novo a rotina ROTU

1.5.2. Determinação do Corte s-t Quando o Algoritmo Parar

Quando o algoritmo termina, com a verificação de que para o fluxo f obtido não existem mais caminhos de acréscimo de fluxo, o conjunto X dos nós que estão rotulados representará exatamente o conjunto de nós atingíveis a partir da fonte em G_f e portanto $(X,\overline{X})$ será um corte mínimo de acordo com o que foi feito na demonstração do Teorema 1.

O conjunto de ramos do corte $(X,\overline{X})$ pode então ser facilmente determinado, por exemplo, re-expandindo os nós de X e verificando se seus sucessores estão ou não em X.

1.5.3. Convergência Finita do Algoritmo

A convergência finita do Algoritmo de Ford e Fulkerson para o caso de capacidades inteiras partindo-se de uma solução inicial viável também inteira, deriva do fato de que os acréscimos de fluxo por ele executados serem então sempre inteiros e portanto maiores ou iguais a um. No caso geral em que podem aparecer capacidades irracionais, essa propriedade de terminação finita se perde conforme mostra o exemplo abaixo:

Construimos uma rede $G = (V,E,c)$ constituida por oito nós: s, x_1, x_2, x_3, y_1, y_2, y_3, t e com três arcos especiais e_1, e_2, e_3 tais que:

$$e_1 = (x_1,y_1) \quad \text{sendo que} \quad c(e_1) = 1$$
$$e_2 = (x_2,y_2) \quad \text{sendo que} \quad c(e_2) = a$$
$$e_3 = (x_3,y_3) \quad \text{sendo que} \quad c(e_3) = a^2$$

Além desses arcos incluimos na rede os arcos:

$$(x_i,x_j) \text{ e } (y_i,y_j) \quad \text{para} \quad i \neq j, \quad i,j \in \{1,2,3\}$$
$$(s,x_i) \quad \text{e} \quad (y_i,t) \quad \text{para} \quad i \in \{1,2,3\}$$

Finalmente acrescentamos o arco (x_1,y_2) com capacidade M. A rede resultante está representada na figura 1.3.

A todos os ramos não especiais atribuimos capacidade ilimitada. Observemos ainda que as capacidades dos arcos especiais cumprem a propriedade: $1 = a + a^2$, $a > 0$.

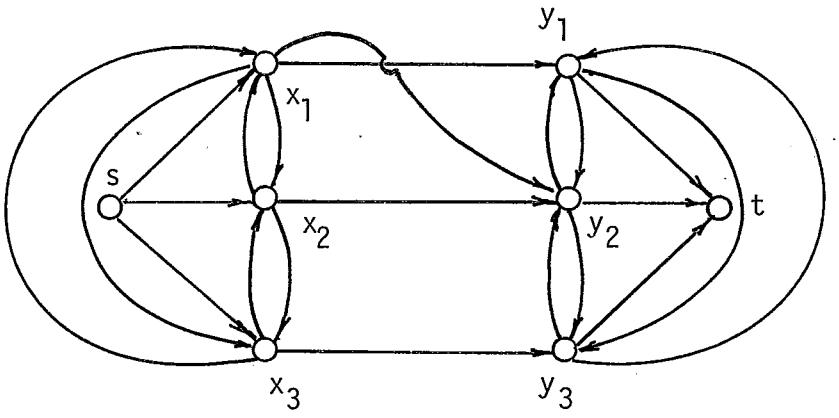


Figura 1.3

Pelas condições $1 = a + a^2$, $a > 0$, temos que $a < 1$ e assim a série $\sum_{n=0}^{\infty} a^n$ converge.

Suponha então que o algoritmo de Ford e Fulkerson parte da situação inicial em que o fluxo é nulo e que na primeira iteração o caminho achado para fazer o acréscimo de fluxo é: (s, x_1, y_1, t).

Feita a atualização dos fluxos teremos que $f(x_1,y_1)=1$ e as capacidades residuais dos três arcos especiais são as representadas na figura abaixo:

$$x_1 \xrightarrow{0} y_1$$

$$x_2 \xrightarrow{a} y_2$$

$$x_3 \xrightarrow{a^2} y_3$$

Figura 1.4

A partir desse ponto consideramos que o algoritmo irá utilizar na iteração n + 1 o caminho: $(s, x_i^n, y_i^n, y_j^n, x_j^n, x_k^n, y_k^n, t)$ onde:

(x_i^n, y_i^n) é o arco especial de menor capacidade residual não nula ao final da iteração n.

(x_j^n, y_j^n) é o arco especial de capacidade residual nula ao final da iteração n. (25)

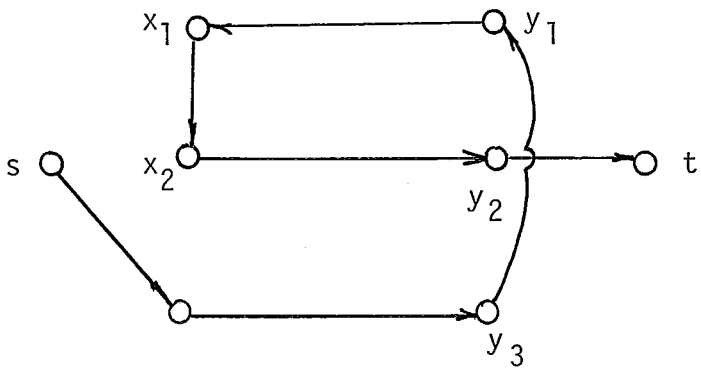
(x_k^n, y_k^n) é o arco especial de maior capacidade ao final da iteração n.

Então por exemplo o segundo e terceiro caminhos de acréscimo de fluxo utilizados seriam:

$(s, x_3, y_3, y_1, x_1, x_2, y_2, t)$

$(s, x_2, y_2, y_3, x_3, x_1, y_1, t)$

respectivamente e que apresetamos nas figuras a seguir:



o caminho: $(s, x_3, y_3, y_1, x_1, x_2, y_2, t)$

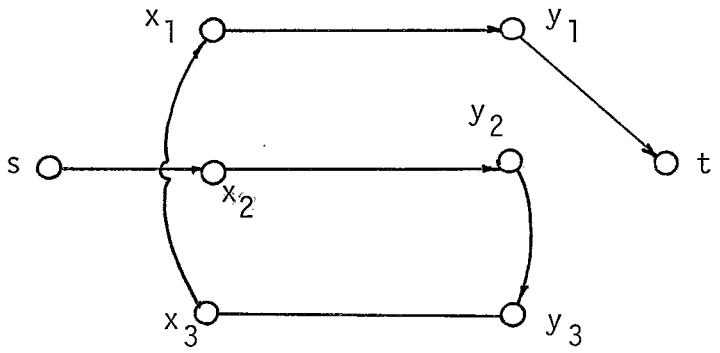
o acréscimo: $f = a^2$

as novas capacidades: $c'(e_1) = a^2$

$c'(e_2) = a - a^2 = a^3$

$c'(e_3) = 0$

Figura 1.5



o caminho: $(s, x_2, y_2, y_3, x_3, x_1, y_1, t)$

o acréscimo: a^3

as novas capacidades: $c''(e_1) = a^2 - a^3 = a^4$

$c''(e_2) = a^3 - a^3 = 0$

$c''(e_3) = a^3$

Figura 1.6

Considere também que, generalizando o que observamos ao final da primeira iteração, tenhamos ao final da iteração $n (n > 1)$ que ocorrem as seguintes capacidades residuais para os arcos especiais

$$\begin{aligned} \delta(x_i^n, y_i^n) &= a^{n+1} \\ \delta(x_j^n, y_j^n) &= 0 \\ \delta(x_k^n, y_k^n) &= a^n \end{aligned} \tag{26}$$

onde $x_i, y_i, x_j, y_j, x_k, y_k$ são definidos em (25).

Vamos verificar que substituindo n por n + 1 em (26) teremos as capacidades residuais dos arcos especiais na iteração seguinte. Para isso observe que o acréscimo realizado através do $(n + 1)^0$ caminho será então de

$$a^{n+1} = \text{mim } \{\infty,\ a^{n+1},\ c(x_j^n,\ y_j^n) > a^2,\ a^n,\ \infty\}$$

e nesse caso obteremos para os ramos especiais as novas capacidades residuais especificadas abaixo:

$$\begin{aligned} \delta(x_i^n, y_i^n) &= a^{n+1} - a^{n+1} = 0 \\ \delta(x_j^n, y_j^n) &= 0 + a^{n+1} = a^{n+1} \\ \delta(x_k^n, y_k^n) &= a^n - a^{n+1} = a^n(1-a) = a^n \cdot a^2 = a^{n+2} \end{aligned} \tag{27}$$

teremos então que

$$\begin{aligned} (x_i^n, y_i^n) &\longleftarrow (x_j^{n+1}, y_j^{n+1}) \\ (x_j^n, y_j^n) &\longleftarrow (x_k^{n+1}, y_k^{n+1}) \\ (x_k^n, y_k^n) &\longleftarrow (x_i^{n+1}, y_i^{n+1}) \end{aligned} \tag{28}$$

Observando conjuntamente (27) e (28) verificamos que é possível estender para $n + 1$ as equações de (26) que portanto serão verdadeiras para todo n natural.

Isso em particular significa que numa iteração n qualquer ($n > 1$) o acréscimo de fluxo segundo o número do caminho será de a^n e que portanto o algoritmo não terá terminação finita.

O fluxo total obtido será de

$$S = 1 + a^2 + a^3 + \ldots + a^n + \ldots \leq \sum_{n=0}^{\infty} a^n < \infty$$

Escolhendo M suficientemente grande podemos fazer o valor do fluxo máximo

$$M + 1 + a + a^2 > S$$

Assim não é possível garantir ao algoritmo de Ford e Fulkerson terminação finita e nem mesmo assegurar que a sequên

cia de valores de fluxo por ele gerada converge para o valor do fluxo máximo. O exemplo acima entretanto dá a impressão que a ciclagem dos caminhos de acréscimo de fluxo que o algoritmo gera se devem ao fato do processo de determinação desses caminhos ser extremamente "mal-intencionado" no sentido de forçar a não-convergência. Ocorre porém que mesmo utilizando critérios em princípio razoáveis para a obtenção desses caminhos pode ocorrer essa ciclagem, como acontece, por exemplo, com a versão do algoritmo de Ford e Fulkerson denominada "Capacity" em que a cada iteração se busca o caminho de maior acréscimo para o fluxo já obtido |[12]|.

No entanto, regras simples como a de procurar a cada iteração o caminho de acréscimo de fluxo entre s e t com menor número de arcos proposta por Edmonds e Karp |[9]| garante a convergência finita do algoritmo conforme veremos numa próxima seção.

1.5.4. Complexidade do Algoritmo

Se m é o número de arcos em E, se F é o valor do fluxo máximo e se as capacidades são inteiras segundo as condições do problema, na tentativa de estimar a complexidade do algoritmo, observamos que:

- o número de acréscimos será no máximo F.
- na atribuição dos rótulos o algoritmo faz uma pesquisa no máximo 2m arcos, pois cada um dos arcos da rede pode ser tomado segundo os dois sentidos.

Logo a complexidade será de, no máximo, O(mF) podendo-se com facilidade gerar uma família de redes R(m,F) onde o número de operações efetuadas pelo algoritmo possa ser realmente, proporcional a esse produto. Para isso basta fazer inicialmente R(m,F) ser a rede da figura abaixo:

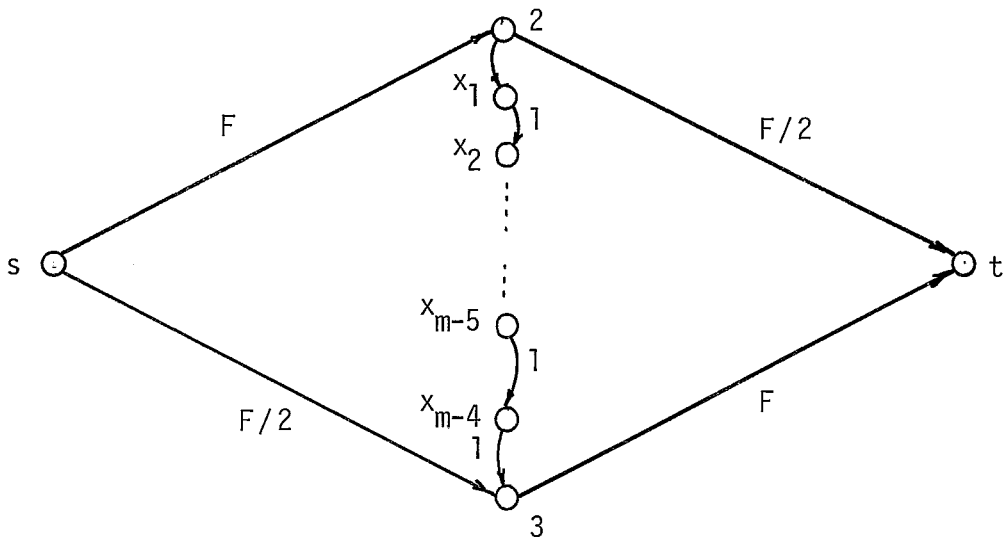


Figura 1.7

Suponha que os dois primeiros caminhos escolhidos pelo algoritmo para fazer o acréscimo de fluxo são, respectivamente:

$$L_1 = (s, 2, x_1, x_2, \ldots, x_{m-5}, x_{m-4}, 3, t)$$

$$L_2 = (s, 3, x_{m-4}, x_{m-5}, \ldots, x_2, x_1, 2, t)$$

ambos permitindo um acréscimo de 1 e que daí em diante o algoritmo os utiliza alternadamente fazendo sempre acréscimos de 1. Ao final, terão sido usado 2.F/2 = F c.a.f. e como cada um deles contêm m-2 ramos o número total de operações efetuadas será proporcional a (m-2).F.

No exemplo anterior verificamos que mesmo ligando diretamente os nós 2 e 3 passando a ter uma rede com cinco ramos, a versão de Ford e Fulkerson com uma regra de expansão que determina os c.a.f. conforme especificamos acima necessitará de exatamente F deles. Se F for muito alto a convergência do algoritmo será então extremamente lenta.

Ao contrário, se os c.a.f. são determinados segundo o procedimento recomendado por Edmonds e Karp e descrito na próxima seção o fluxo máximo em qualquer rede capacitada do tipo apresentada na figura 1.7 será obtido usando-se apenas dois caminhos de acréscimo de fluxo.

1.6. ALGORITMO DE EDMONDS E KARP (ALGORITMO E-K)

1.6.1. O Algortimo

Na parte final da seção anterior, no exemplo da figura 1.7 os caminhos de acréscimo de fluxo que escolhemos tem comprimento igual a m-2. Existem entretanto dois caminhos de acréscimo de fluxo independentes, de comprimento apenas dois cuja utilização nos leva imediatamente ao fluxo máximo achado.

Uma forma de determinar os caminhos de acréscimo de de fluxo que leva a essa escolha foi proposta por Edmonds e Karp em 1969 na "Calgary International Conference on Combinatorial Structures and their Applications" e divulgada em 1972 pela J.A.C.M. |9|. O que eles propuseram foi basicamente no algoritmo F-F tornar mais específico o processo de escolha do nó rotulado a ser expandido empregando para isso a regra FIFO da busca horizontal |8|.

Em termos de se obter a versão de Edmonds e Karp do algoritmo de Ford e Fulkerson tudo que precisamos fazer é indicar no Passo 1 da rotina ROTU que o nó ainda não expandido que deve ser escolhido é aquele que for gerado há mais tempo. Fazendo sempre isso teremos que cada caminho de acréscimo de fluxo entre s e t utilizado, será sempre um caminho com menor número de arcos entre s e t na rede residual em que ele é calculado. Veremos na seção seguinte que o número de arcos desses caminhos jamais decresce tendo obrigatoriamente que crescer se voltamos a saturar um ramo ou eliminar inteiramente seu fluxo. Isso leva a convergência finita do algoritmo.

1.6.2. Convergência e Complexidade do Algoritmo

O resultado principal de onde pode ser extraída a convergência finita do método e sua ordem de complexidade é o seguinte:

TEOREMA - Sejam $\sigma_i^{(k)}$ e $\tau_i^{(k)}$ definidas como abaixo.

$\sigma_i^{(k)} \equiv$ o mínimo número de arcos num c.a.f. da fonte s até o nó i, depois de ter feiro k acréscimos de fluxo. (29)

$\tau_i^{(k)} \equiv$ o mínimo número de arcos num c.a.f. do nó i até o poço t, depois de ter feito k acréscimos de fluxo. (30)

Então, se verifica que se cada acréscimo de fluxo é feito num c.a.f. com o menor número de arcos possível,

$$\sigma_i^{(k+1)} \geq \sigma_i^{(k)} \tag{31}$$

$$\tau_i^{(k+1)} \geq \tau_i^{(k)} \tag{32}$$

Prova

Suponhamos que existe i e k tais que $\sigma_i^{(k+1)} < \sigma_i^{(k)}$, seja especificamente

$$\sigma_i^{(k+1)} = \min_j \{\sigma_j^{(k+1)} / \sigma_j^{(k+1)} < \sigma_j^{(k)}\} \tag{33}$$

Se lembramos que $\sigma_s^{(i)} = 0, \forall_i$ então

$$\sigma_i^{(k+1)} \geq 1.$$

Seja (i,j) ou (j,i) o último arco no menor c.a.f. de s até i depois do (k + 1)-ésimo acréscimo. Suponhamos que o arco dito é (j,i), um arco direto com a característica de ter $f(j,i) < c(j,i)$, então $\sigma_i^{(k+1)} = \sigma_j^{(k+1)} + 1$ e pela definição (33) temos

$$\sigma_i^{(k+1)} \geq \sigma_j^{(k)} + 1$$

Depois do k-ésimo acréscimo deveriamos ter $f(j,i) = c(j,i)$ pois caso contrário teríamos

$$\sigma_i^{(k)} \leq \sigma_j^{(k)} + 1 \leq \sigma_i^{(k+1)} \tag{34}$$

contradizendo nossa suposição.

Mas, se $f(j,i) = c(j,i)$ depois do k-ésimo acréscimo e $f(j,i) < c(j,i)$ depois do $(k + 1)$-ésimo acréscimo teremos que (j,i) é um arco contrário no $(k + 1)$-ésimo caminho de acréscimo de fluxo. Portanto, se o caminho tem o menor número de arcos, temos que

$$\sigma_j^{(k)} = \sigma_i^{(k)} + 1$$

e como em (34)

$$\sigma_j^{(k)} + 1 \leq \sigma_i^{(k+1)},$$

segue-se que

$$\sigma_i^{(k+1)} + 2 \leq \sigma_i^{(k+1)}$$

contrariando nossa suposição. Logo segue que

$$\sigma_i^{(k+1)} < \sigma_i^{(k)}$$

é falsa, portanto

$$\sigma_i^{(k+1)} \geq \sigma_i^{(k)}$$

completando-se assim a prova de (31).

Uma argumentação inteiramente análoga levaria a prova (32) e portanto

$$\tau_i^{(k+1)} \geq \tau_i^{(k)}$$

C.Q.D.

Antes de apresentarmos um corolário desse teorema que praticamente definirá a convergência do método, é importante esclarecer que quando dizemos que um arco (i,j) é saturado (ou dessaturado) pelo algoritmo numa dada iteração i, se:

1) (i,j) está na rede residual do fluxo obtido até a iteração anterior (G_i') e o fluxo em (i,j) se torna igual (ou deixa de ser, respectivamente) a sua capacidade após a i-ésima iteração.

2) (j,i) está em G_i' e o fluxo em (i,j) se torna igual a zero (deixa de se anular, respectivamente) após a i-ésima iteração.

Feito isso podemos enunciar o corolário citado de forma mais compacta assim:

Corolário

Se um arco é saturado na k-ésima e q-ésima iterações com q > k, então

$$\gamma^{(q)} \geq \gamma^{(k)} + 2$$

onde $\gamma^{(i)}$ é o número de arcos contidos no i-ésimo caminho de acréscimo de fluxo utilizado pelo algoritmo.

Prova

Se o ramo (i,j) é saturado na k-ésima e na q-ésima iterações, então existe p tal que k < p < q e o arco (i,j) é dessaturado na p-ésima iteração. Nessa iteração (i,j) é portanto percorrido num sentido diferente do que foi na k-ésima e des

se modo

$$1 + \sigma_j^{(p)} = \sigma_i^{(p)} \tag{35}$$

Por outro lado, pelo teorema anterior temos que:

$$\sigma_j^{(p)} \geq \sigma_j^{(k)} \tag{36}$$

e além disso

$$\sigma_j^{(k)} = \sigma_i^{(k)} + 1 \tag{37}$$

juntando (35), (36) e (37), temos

$$\sigma_i^{(p)} \geq \sigma_i^{(k)} + 2 \tag{38}$$

Como, de novo, pelo teorema temos

$$\sigma_i^{(q)} \geq \sigma_i^{(p)} \tag{39}$$

$$\tau_i^{(q)} \geq \tau_i^{(k)} \tag{40}$$

de (39) e (38) temos:

$$\sigma_i^{(q)} \geq \sigma_i^{(k)} + 2 \tag{41}$$

De (40) e (41) temos:

$$\sigma_i^{(q)} + \tau_i^{(q)} \geq \sigma_i^{(k)} + \tau_i^{(k)} + 2$$

e, portanto:

$$\gamma^{(q)} \geq \gamma^{(k)} + 2$$

C.Q.D.

Em consequência, do resultado anterior, e como $\gamma^{(i)} \leq n$, $\forall\, i \in N$, cada arco só pode ser saturado em no máximo n/2 iterações. Como em cada iteração um arco é saturado, temos que o número de iterações fica então limitado por n/2 . m. Além disso observe que em cada iteração, tanto o processo de busca efetuado pelo algoritmo para achar o caminho disponível de menor número de arcos entre s e t, como a própria atualização dos fluxos nos ramos desse caminho podem ser feitos num tempo de O(m). Desse modo o número total de operações efetuado pelo método de Edmonds e Karp é no máximo de

$$O(m \cdot \frac{n}{2} \cdot m) = O(m^2 \cdot n) \ .$$

CAPÍTULO 2

ALGORITMO DE E.A. DINIC

2.1. INTRODUÇÃO

O Algoritmo proposto por Edmonds e Karp, sendo uma versão do algoritmo de Ford e Fulkerson realiza uma expansão a partir da fonte para achar cada caminho de acréscimo de fluxo que ele utiliza, e que tem a propriedade de conter o menor número de arcos entre todos os caminhos da fonte ao poço na rede residual.

Fazer todas essas expansões significa em geral, repetir uma série de operações pois de uma iteração para a seguinte podemos retirar da rede residual apenas um único arco. Uma idéia para acabar com todas essas repetições é construir um outro grafo auxiliar constituído pelos nós e arcos da rede residual que façam parte de algum caminho entre a fonte e o poço com número mínimo de ramos. A capacidade desses ramos no novo grafo que chamaremos de Referentes é mantida a mesma da rede residual.

Obtido o referente que será um grafo em camadas onde a primeira contém apenas a fonte e a última só o poço e onde qualquer nó estará num caminho de acréscimo de fluxo entre esses dois (1), procedemos da seguinte forma:

A partir de um fluxo identicamente nulo determinamos um caminho de acréscimo de fluxo qualquer entre s e t, e atualizamos o fluxo em seus ramos. Eliminamos em seguida os nós e ramos que resultarem bloqueados em função desse acréscimo. Após

fazer isso poderemos garantir que ou bloqueamos a fonte e nesse caso não há mais c.a.f. no referente ou então, como inicialmente tínhamos (1),fazendo uma simples busca em profundidade sem "backtraking" poderemos determinar um novo c.a.f. Esse processo de busca é consideravelmente mais simples do que a busca horizontal que efetuávamos entre um c.a.f. e outro no algoritmo E-K. e isso determinará uma complexidade menor para esse novo método.

Quando no referente não se puder mais encontrar nenhum c.a.f. então alteramos o fluxo nos ramos do grafo original, calculamos a nova rede residual e a partir dele um novo referente. Repetimos então o procedimento acima e assim por diante até obtermos um fluxo máximo. Com respeito a esse novo referente se vai poder garantir que ele obrigatoriamente terá mais uma camada do que o anterior, ou seja, a distância entre a fonte e o poço na rede residual irá crescer.

Esse resultado acarreta que a exemplo do que ocorre com Edmonds e Karp a sequência de caminhos de acréscimo de fluxo usados estará ordenada de forma não decrescente segundo o número de seus ramos. E o que é mais importante, ele irá limitar o número de referentes que precisam ser construídos o que irá determinar a convergência finita do método.

Um algoritmo concebido segundo essas idéias gerais é o de E.A. Dinic apresentado em 1970 |3| na União Soviética e do qual passamos agora a definir formalmente os procedimentos básicos e os principais elementos utilizados.

2.2. O REFERENTE

2.2.1. Definição

Sejam a rede capacitada G = (V,E,c), o fluxo f na rede, a rede residual G' = (V,E',c') e k o menor comprimento de um caminho da fonte ao poço em G', chama-se Referente R à reunião dos caminhos de acréscimo de fluxo de comprimento k entre a fonte s e o poço t.

Formalmente se $\hat{V}$ representa o conjunto de nós de R e se $\hat{E}$ é o conjunto de seus ramos, temos:

$$v \in \hat{V} \iff v \in V', \quad d(s,v) + d(v,t) = k \tag{1}$$

$$e = (u,v) \in \hat{E} \iff e \in E', \quad d(s,u) + 1 = d(s,v) \tag{2}$$

As expressões (1) e (2) acima, permitem, então, particionar o conjunto de nós de R de modo de fazer dele um grafo em camadas conforme indicamos abaixo:

$$\hat{V} = \bigsqcup_{i=0}^{k} V_i \tag{3}$$

onde $V_i = \{v / v \in \hat{V}, \quad d(s,v) = i\}$

$$\hat{E} = \bigsqcup_{i=1}^{k} E_i \tag{4}$$

onde $E_i = \{e = (u,v) / e \in \hat{E}, \quad u \in V_{i-1}, \quad v \in V_i\}$

A figura abaixo esclarece as idéias apresentadas.

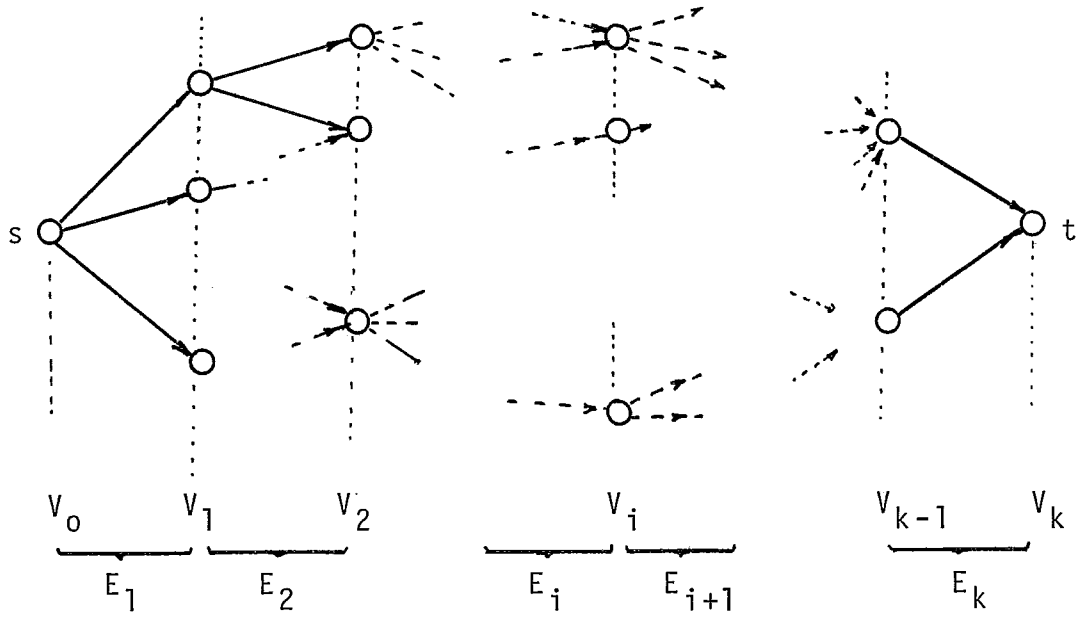


Figura 2.1

2.2.2. Obtenção do Referente

Considerando a definição 2.2.1, o procedimento para obter o referente $R = (\hat{V},\hat{E},c')$ a partir do grafo original pode ser o apresentado a seguir, em que basicamente se efetua uma busca horizontal onde se vai construindo parte da rede residual, inclusive, calculando a capacidade de seus ramos e estruturando-a em camadas, segundo a distância à fonte, até que o poço seja atingido. Uma Programação Dinâmica de trás para a frente, é então efetuada para eliminar os nós e ramos que não farão parte do Referente.

ALGORITMO PARA OBTENÇÃO DO REFERENTE

Dados $G = (V,E,c), s, t$, $\forall v$: $T(v) = \{$ramos adjacentes a $v\}$ e um fluxo f de s a t,

faça:

Para todo $v \in V$, $\overline{A}(v) = \emptyset$

$V_0 = \{s\}$, $i = 0$ e

Enquanto $t \notin V_i$ e $V_i \neq \emptyset$, execute:

$V_{i+1} = \emptyset$

Para cada nó (v) de V_i, execute:

Faça $\overline{S}(v) = \emptyset$

Para todo $r \in T(v)$, execute:

v' = outro nó adjacente a r

Se $v' \notin \bigcup_{k=0}^{i} V_k$, execute:

Se $r = (v,v')$ e $f(r) < c(r)$

Faça $\overline{S}(v) = \overline{S}(v) \cup \{r\}$

$\overline{A}(v') = \overline{A}(v') \cup \{r\}$

$c'(r) = c(r) - f(r)$

$V_{i+1} = V_{i+1} \cup \{v'\}$

Se $r = (v',v)$ e $f(r) > 0$

Faça $r' = (v,v')$

$\overline{S}(v) = \overline{S}(v) \cup \{r'\}$ e

$\overline{A}(v') = \overline{A}(v') \cup \{r'\}$

$c'(r') = f(r)$

$V_{i+1} = V_{i+1} \cup \{v'\}$

Fim-Se

Fim-Para

Fim-Para

$i = i + 1$

Fim-Enquanto

Se $t \in V_i$, execute:

$k \leftarrow i$, $V_k = \{t\}$

Enquanto $i > 0$, faça

$E_i = \emptyset$

Para cada $v \in V_{i-1}$ faça

Para cada $r = (v,v') \in \overline{S}(v)$, execute:

Se $v' \notin V_i$, faça

$\overline{S}(v) = \overline{S}(v)/\{r\}$

Fim-Para

Se $\overline{S}(v) \neq \emptyset$, faça $E_i = E_i \cup \overline{S}(v)$

senão, faça $V_{i-1} = V_{i-1}/\{v\}$

Fim-Para

Fim-Enquanto

Faça: $\hat{E} = \bigcup_{i=1}^{k} E_i$, e

$\hat{V} = \bigcup_{i=0}^{k} V_i$

Fim-Se

Se $V_i = \emptyset$, pare, pois não existe caminho de acréscimo de fluxo e portanto o fluxo é ótimo.

Fim

Comentando suscintamente o algoritmo acima, devemos dizer que no "loop" do primeiro "enquanto", ele efetua a citada busca horizontal a partir da fonte para determinar o caminho de acréscimo de fluxo de menor tamanho entre s e t. No desenvolvimento dessa busca ele vai compondo as camadas V_i do referente e para cada nó v atingido ele determina os conjuntos $\overline{S}(v)$ e $\overline{A}(v)$ que representam os arcos que ligam v a um nó distando da fonte na rede residual mais um arco e menos um arco respectivamente.

De $\overline{S}(v)$ o algoritmo, construirá as camadas de arcos (E_i) do referente.

Quando essa primeira fase é completada temos em cada V_i, i = 1,2,...,k o conjunto de nós para os quais existe um caminho de acréscimo de fluxo, a partir da fonte, com i arcos. Entretanto, nem todos esses nós farão parte do referente pois não necessariamente existirá um c.a.f. com k-i arcos entre um deles e o poço. Será necessário então eliminá-los bem como os arcos a eles adjacentes. Essa eliminação é efetuada no "loop" do segundo enquanto em que o algoritmo percorre as camadas V_i em ordem decrescente mas a cada nó v atingido encontra todos os seus sucessores exatamente como faz a programação dinâmica de trás para a frente. Se um desses sucessores não está mais em V_{i+1} o arco que o liga a v é retirado de $\overline{S}(v)$ e se durante esse processo $\overline{S}(v)$ se tornar vazio o próprio v sai de V_i. Se ao contrário, ao final do tratamento do nó v ainda houver elementos em $\overline{S}(v)$, esse conjunto será incorporado a E_i.

Quando nesse segundo "loop" atingimos a fonte, a construção do referente estará completa e passaremos a tentar obter nele um fluxo maximal. Se entretanto no decorrer do processo de busca horizontal geramos uma camada V_i vazia então podemos mesmo parar o algoritmo pois nesse caso não haverá mais nenhum c.a.f. entre a fonte e o poço e o fluxo será máximo.

Passamos agora a abordar o procedimento empregado pelo método para a obtenção de um fluxo maximal no referente.

2.2.3. Determinação de um Fluxo Maximal no Referente

Conforme vimos, o algoritmo dado em 2.2.2. obtém o referente (R) já decomposto em suas camadas de nós (V_i) e arcos (E_i). Ele determina também para cada nó (v) de R o conjunto de seus arcos sucessores $\overline{S}(v)$ e antecessores $\overline{A}(v)$ bem como para cada arco r do referente sua capacidade (c'(r)) e seus nós de chegada e saída (a(r), s(r)). Da posse desses elementos, poderemos utilizar para obter um fluxo maximal em R o algoritmo abaixo que comentaremos a seguir.

Dados $(V_i;\ i = 1,\ldots,k)$; $(E_i;\ 1,\ldots,k)$;

$(\overline{A}(v)\ ,\ \forall v \in \bigcup_{i=1}^{k} V_i)$; $(\overline{S}(v),\ \forall v \in \bigcup_{i=0}^{k-1} V_i)$;

$(a(r),\ s(r),\ c'(r),\ \forall r \in \bigcup_{i=1}^{k} E_i)$; s e t, faça

Enquanto S(s) $\neq \emptyset$, execute:

v = s

i = 0

Enquanto v $\neq$ t, execute:

Escolha r $\in \overline{S}(v)$

Faça i = i + 1

h(i) = r

$\hat{c}$ = mim $\{\hat{c}, c'(r)\}$

v = s(r)

Fim-Enquanto

A

Para cada i decrescendo de k até 1, execute:

$\overline{r}$ = h(i)

$c'(\overline{r}) = c'(\overline{r}) - \hat{c}$

$f'(\overline{r}) = f'(\overline{r}) + \hat{c}$

Se $c'(\overline{r}) = 0$, faça:

B

p = a($\overline{r}$) | C

$\overline{S}$(p) = $\overline{S}$(p)/{$\overline{r}$}

Para todo v ∈ V_{i-1}, execute:

Se $\overline{S}$(v) = ∅ execute | D

Para todo r ∈ $\overline{A}$(v), faça:

v' = a(r)

$\overline{S}$(v') = $\overline{S}$(v')/{r}

Fim-Para

V_{i-1} = V_{i-1}/{v}

Fim-Se

Fim-Para

Fim-Para

Fim-Enquanto

Fim

No procedimento indicado por A o algoritmo constrói um c.a.f. sem necessidade de nenhum "backtracking" e em B ele efetua o acréscimo de fluxo, que é o máximo possível, nos arcos desse caminho.

Feito esse acréscimo, pelo menos um dos arcos do caminho fica saturado e é eliminado do referente em C. Além disso quando saturamos um arco r = (u,v), podemos criar "becos sem saída", isto é, pode haver arcos onde, mesmo sendo o fluxo menor que a capacidade, qualquer acréscimo que pretendamos fazer passar por eles, implica num aumento de fluxo no arco em questão o que não é mais possível. Essa é por exemplo a situação dos arcos $e_1 = (s,n_1)$, $e_2 = (n_1,n_3)$, $e_3 = (n_2,n_3)$ com respeito ao arco $e_4 = (n_3,t)$ saturado na rede da figura 2.3.

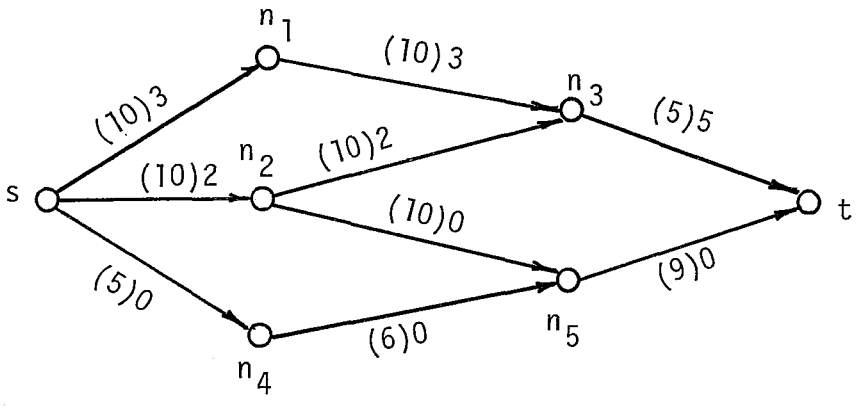


Figura 2.3

Os arcos nessa situação ditos bloqueados, bem como os nós que tem todos os arcos sucessores com essas características são eliminados em D.

Todo esse processo vai sendo repetido obtendo-se novos c.a.f., atualizando o fluxo em seus arcos e eliminando os elementos bloqueados até que da própria fonte não saia mais nenhum arco. Quando isso acontecer, logicamente, s estará bloqueado e voltamos a usar o algoritmo da seção 2.2.2. para tentar obter um novo referente.

Passamos agora a estudar a complexidade do algoritmo de Dinic.

2.3. CONVERGÊNCIA E COMPLEXIDADE DO ALGORITMO

A convergência finita do algoritmo está garantida pelas mesmas condições que asseguram a terminação do método de Edmonds e Karp dado que ele também só utiliza caminhos de acréscimo de fluxo como o menor número de arcos possível. Para mostrar

que a complexidade em tempo do algoritmo se torna menor que o de Edmonds e Karp precisamos inicialmente do seguinte resultado.

LEMA D-1

Quando acabamos de processar um referente com k camadas não existem mais caminhos de acréscimo de fluxo com k ramos ou menos.

Prova

Suponha que isso não seja verdade e que possa existir um referente R com k camadas e tal que chamando de f e $\overline{f}$ o fluxo na rede original antes e depois do seu processamento, temos um caminho de acréscimo de fluxo L em $G'_{\overline{f}}$ de tamanho menor que k e que portanto não pode constar de G'_f. Mas para que L que não pertence a G'_f venha a aparecer depois em $G'_{\overline{f}}$ é necessário que para algum de seus arcos (i,j), (j,i) pertença a G'_f e mais precisamente que (j,i) tenha sido usado durante o tratamento de R por um caminho de acréscimo de fluxo L'.

Entretanto da demonstração do corolário do teorema na seção 1.6.2., temos como "lay-product" a seguinte afirmação:

"Se usamos sempre c.a.f. de menor número de arcos, então depois que um c.a.f. P contendo o arco (j,i) é empregado qualquer c.a.f. passando por (i,j) terá tamanho maior que P".

Fazendo então P = L' nessa afirmação e representando por $\gamma(L)$ o número de arcos de cada caminho L, obtemos

$$k = \gamma(L') < \gamma(L)$$

contrariando nossa hipótese inicial.

C.Q.D.

Como consequência imediata deste lema temos que o algoritmo constrói e processa não mais de (n-1) referentes e que portanto se ele gasta $O(\tau)$ com cada referente, sua complexidade será no máximo $O(n\tau)$.

Passamos agora a tentar determinar τ.

2.3.1. Tempo Gasto na Construção do Referente

O processo de construção do referente é constituido basicamente por uma busca horizontal para determinar o caminho de menor número de arcos entre s e t (1) e uma programação dinâmica (2) para eliminar nós e ramos que foram atingidos em (1) mas que não farão parte do referente.

Ambos esses processos são sabidamente O(m) dado que cada arco é tratado uma única vez em qualquer deles. Desse modo a construção do referente fica sendo O(m).

2.3.2. Tempo Gasto na Obtenção de um Fluxo Maximal no Referente

No que diz respeito a obtenção de um fluxo maximal no referente, devemos observar o seguinte:

Quando determinamos um c.a.f. ligando a fonte ao poço no referente, um arco, pelo menos é bloqueado e portanto o número desses caminhos é O(m).

Cada caminho possui no máximo n-1 arcos e como tudo que fazemos para determiná-los é "andar para frente" gastamos O(n) para achar cada um deles e portanto O(nm) no total.

Idêntico tempo é dispensado no trabalho de atualização do fluxo nos ramos de cada c.a.f. mas na eliminação de arcos já bloqueados o tempo total é de O(m) que, é claro, cada arco só é eliminado uma vez. Finalmente devido ao fato de termos O(m) c.a.f. o número de vezes em que durante a execução entre um c.a.f. e outro, o algoritmo pergunta se um nó está bloqueado é O(nm). Juntando todos esses resultados podemos verificar que a complexidade em tempo do método de Dinic na obtenção de um fluxo maximal no referente é O(nm).

Finalmente reunindo as conclusões de 2.3.1. e 2.3.2. verificamos que podemos fazer $\tau = \max\{m,nm\} = nm$ e que portanto o algoritmo de Dinic será $O(n^2m)$.

CAPÍTULO 3

ALGORITMO DE A.V. KARZANOV

3.1. INTRODUÇÃO

Na procura da solução do Problema de Fluxo Máximo, as abordagens já descritas foram sucessivamente sendo melhoradas em termos do tempo gasto na busca do caminho de acréscimo de fluxo.

O algoritmo E-K é uma versão do algoritmo F-F com convergência garantida pelo fato dele procurar, sempre, usar o caminho de acréscimo de fluxo mais curto entre a fonte e o poço. O de Dinic faz uso de um subgrafo da rede residual o qual, está garantido, contém todos os caminhos de acréscimo de fluxo de menor comprimento entre s e t e que vão sendo sucessivamente bloqueados com uma simples busca em profundidade sem "volta".

Mas até aqui, como sempre, utilizamos c.a.f. para fazer as alterações de fluxo, podemos dizer que sempre dispomos de um fluxo viável em qualquer momento da execução do algoritmo.

Entretanto é possível, pelo menos no que se refere a obtenção de um fluxo maximal no referente, obter um algoritmo mais eficiente abandonando a utilização de c.a.f. e com isso a garantia de se ter sempre um fluxo. Isso foi o que mostrou A.V. Karzanov [6] com seu método, o qual chamaremos de Algoritmo K.

Esse algoritmo é idêntico ao de Dinic, exceto na metodologia empregada para bloquear a fonte no referente e por isso limitamos nossa apresentação ao tratamento desse problema. Essa

metodologia que diminui para $O(n^2)$ os $O(nm)$ gastos por Dinic para efetuar essa operação, pode ser melhor descrito usando-se o conceito de Prefluxo que introduziremos a seguir junto com outros elementos utilizados pelo algoritmo K.

3.2. DEFINIÇÕES BÁSICAS

3.2.1. Função Admissível

Seja e um arco numa rede capacitada G = (V,E,c), chama-se de Função Admissível a aquela função real h cujo domínio é E e tal que $0 \leq h(e) \leq c(e)$, para todo $e \in E$.

3.2.2. Nó Deficiente ou Não Equilibrado

Se h é uma função admissível, $v \in V$ e um nó numa rede G = (V,E,c), chama-se de Nó Deficiente aquele nó que tem a seguinte propriedade:

$$D_h(v) = \sum_{u \in A(v)} h(u,v) - \sum_{w \in S(v)} h(v,w) \neq 0$$

3.2.3. Deficiência do Nó

O valor $D_h(v)$ da definição 3.2.2. é chamado a deficiência do nó v com relação a h.

Sempre que não houver risco de confusão omitiremos o

índice h para simplificar a notação.

3.2.4. Nó Equilibrado

Se na definição 3.2.2. temos que $D_h(v) = 0$, o nó v é dito Equilibrado ou não-deficiente.

3.2.5. Prefluxo de s a t

Dados: uma rede capacitada G = (V,E,c) e s e t dois de seus nós, chama-se de Prefluxo de s a t à função real $g:E \longrightarrow \mathbb{R}$ que tem as seguintes propriedades:

P.1. - $0 \leq g(e) \leq c(e)$
ou seja, g é uma função admissível.

P.2. - $\sum_{x \in A(v)} g(x,v) \geq \sum_{y \in S(v)} g(v,y), \quad \forall v \in \overline{V}$
isto é, a deficiência de qualquer nó é sempre não negativa.

P.3. - Se $v \in \overline{V}$ é um nó deficiente então o nó v é g-bloqueado.

P.4. - A fonte s é g-bloqueada.

3.3. CARACTERIZAÇÃO DO FLUXO MAXIMAL EM TERMOS DE PREFLUXO

Segundo a definição 3.2.5. acima, se dado um prefluxo g, não existirem nós deficientes, então g será um fluxo maximal. É que nesse caso, sendo equilibrada g será mesmo um fluxo de s a t e por P.4. a fonte estará bloqueada.

É exatamente a caracterização de Fluxo Maximal dada acima que o algoritmo de Karzanov, que passamos a descrever, utiliza para bloquear a fonte no referente.

3.4. A SOLUÇÃO INICIAL

Para a solução do nosso problema, inicialmente, fazemos o fluxo igual a capacidade nos ramos que saem da fonte e zero no resto. Assim, partimos já satisfazendo as condições P.1., P.2. e P.4. que manteremos sempre atendidas durante toda a execução. Não temos, entretanto P.3. pois os nós da primeira camada estão, tanto deficientes como não-bloqueados.

3.5. PROCEDIMENTOS BÁSICOS

O algoritmo busca obter um fluxo maximal fazendo uso da caracterização dada na seção 3.3. e, por isso mesmo, é constituído de dois procedimentos básicos:

1) AVANÇO que visa obter um prefluxo de s a t, ou mais precisamente de satisfazer a condição P.3, já que as outras três condições de definição de prefluxo estarão permanentemente aten

didas.

2) EQUILÍBRIO, que só é aplicado quando a solução vigente já for um prefluxo e objetiva equilibrar os nós deficientes.

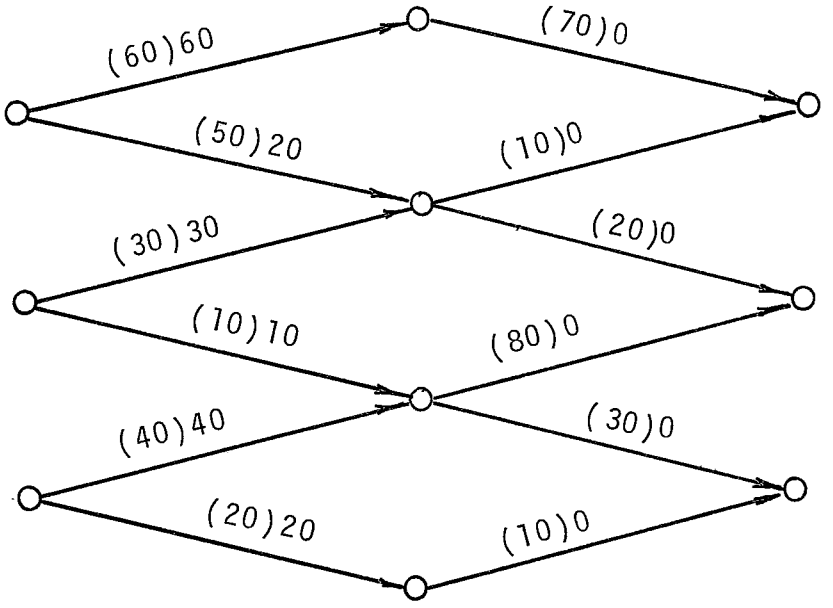
Cada vez que chamamos estes procedimentos, eles tratam conjuntamente todos os nós de uma camada e a ação que eles efetuam nesses nós, é descrita a seguir.

3.6. PROCEDIMENTO DE AVANÇO

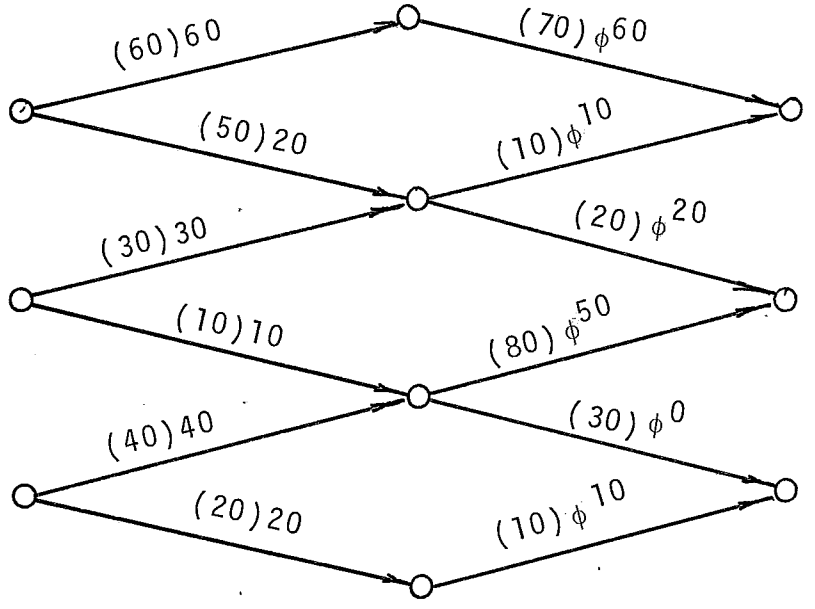
Se um nó v da camada está equilibrado ou bloqueado, Avanço não modifica o fluxo nos arcos a ele adjacentes. Mas se v está deficiente e não-bloqueado então Avanço faz escoar o máximo possível a deficiência de v pelos ramos ainda livres que dele partem. Se a deficiência de v for menor ou igual a soma das folgas desses ramos ela será então, inteiramente suprimida e o nó resultará equilibrado. Se for maior então todas essas folgas serão esgotadas e portanto não haverá mais ramos livres saindo de v que passará a ser bloqueado.

Ao final da aplicação do procedimento a uma camada V_i, onde $i < k$, portanto, se cumprirá P.3. em todos os seus nós. Para isso, entretanto, fez-se chegar um acréscimo de fluxo a nós livres de V_{i+1} que obrigatoriamente estarão deficientes nessa ocasião.

Vejamos um exemplo:



3.1."a"



3.1."b"

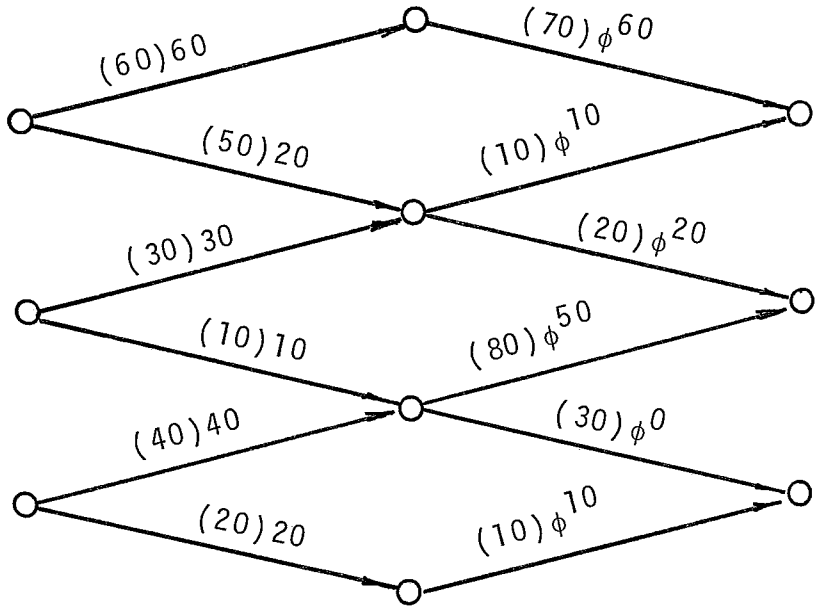
Figura 3.1

3.7. PROCEDIMENTO DE EQUILÍBRIO

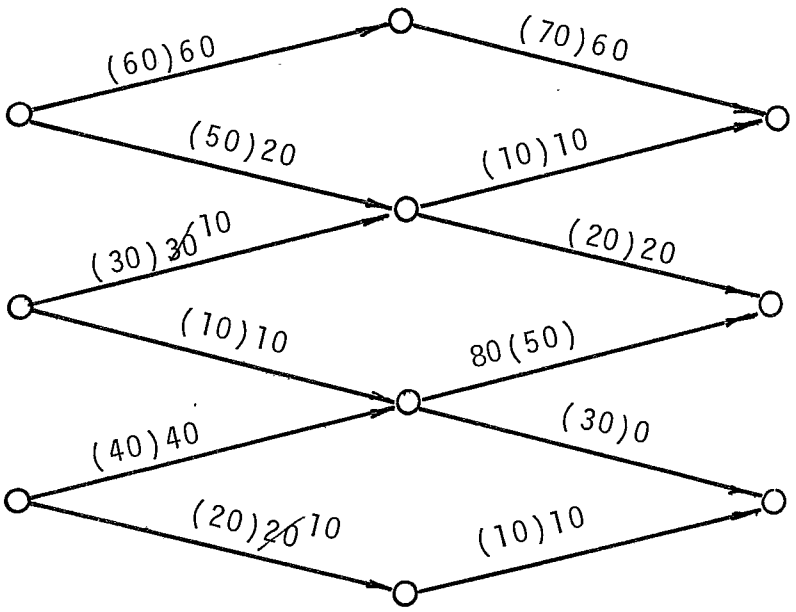
Quando aplicamos Equilíbrio, já temos um prefluxo e portanto não existem nesse momento, nós deficientes e não bloqueados.

Para cada nó v deficiente, e portanto bloqueado, da camada que está tratando, Equilíbrio elimina o excesso nele acumulado reduzindo o fluxo em seus arcos de entrada. A supressão completa de deficiência de v, procedendo-se deste forma, está assegurada pela propriedade P.2 , verdadeira, durante toda a aplicação do algoritmo.

Ao fazer essa redução na entrada de fluxo num nó de uma camada V_i o procedimento faz aumentar, ou cria, deficiências em V_{i-1}. Em particular, ele pode tornar deficiente um nó não-bloqueado de V_{i-1} quando então, deixaremos de ter um prefluxo.



3.2."a"



3.2."b"

Figura 3.2

3.8. CONDIÇÕES MANTIDAS E A SEQUÊNCIA DE PROCEDIMENTOS EMPREGADOS PELO ALGORITMO

Observemos em primeiro lugar que a aplicação de qualquer dos procedimentos a uma camada equilibra seus nós e pode apenas aumentar, mas nunca diminuir a deficiência de nós de outras camadas o que impede a criação de deficiências negativas. Assim P.2 irá se manter durante toda a execução. Como também sempre respeitamos as restrições de capacidade conservaremos P.1 e além disso para dessaturarmos um arco (s,v) durante a aplicação de Equilíbrio a V_1 será necessário que v esteja bloqueado e portanto, a fonte jamais será livre mantendo-se P.4.

Assim, verificamos que a afirmação feita anteriormente de que essas três condições sempre ocorrem é verdadeira.

Vejamos agora como as condições restantes: P.3 e o equilíbrio de todos os nós, são conseguidas.

Representemos agora por A_i e EQ_i, $0 \leq i \leq k$ a aplicação de Avanço e Equilíbrio a camada V_i, respectivamente.

Após a inicialização, como vimos, apenas V_1 não satisfez P.3. Executamos então A_1 para resolver esse problema, mas com isso, fazemos surgir nós deficientes não bloqueados em V_2 e para eliminá-los executamos A_2 e assim sucessivamente até que chegamos ao poço quando então teremos um prefluxo. Obtido esse prefluxo passamos a tentar equilibrá-lo aplicando inicialmente EQ_{k-1} que pode criar deficiências em V_{k-2}. Se essas deficiências não atingem nós livres de V_{k-2}, executamos EQ_{k-2} podendo perturbar o equilíbrio de V_{k-3}. Se apesar dessa perturbação continuamos a ter um prefluxo então fazemos EQ_{k-3} e assim por diante até, supondo que não voltemos à fonte, e que passemos a ter nós

deficiêntes e não bloqueados na camada V_i imediatamente anterior a última a que aplicamos Equilíbrio.

Como no subgrafo formado pelas camadas até V_i os fluxos não foram alterados por Equilíbrio, elas continuam satisfazendo P.3 e desse modo aplicando agora $A_i, A_{i+1}, \ldots, A_k$ voltaremos a conseguir um prefluxo. De novo fazemos EQ_{k-1}, EQ_{k-2}, etc. e no caso de deixarmos de ter um prefluxo voltamos a sequência de Avanços indicada acima e assim por diante até que se obtenha a regra de parada do algoritmo que é indicada a seguir.

3.9. REGRA DE PARADA

Depois de aplicarmos EQ_1 podemos parar o algoritmo porque:

i) Se decidimos aplicar EQ_1 porque tínhamos um prefluxo e a única possibilidade de EQ_1 destruir essa condição era ele criar deficiências em nós livres de $V_0 = \{s\}$. Como a fonte é sempre bloqueada isso é impossível e portanto após EQ_1 continuamos a ter um prefluxo.

ii) Quando executamos EQ_i o fazemos ao final de uma sequência $EQ_{k-1}, EQ_{k-2}, \ldots, EQ_{i+1}, EQ_i$ que equilibra todas as camadas de V_i ao poço.

Fazendo, então $i = 1$ teremos que após EQ_1 todos os nós do grafo estarão equilibrados.

Juntando (i) e (ii) verificamos que a solução corrente após a execução de EQ_1 é um prefluxo equilibrado o que caracteriza um fluxo maximal.

3.10. O ALGORITMO

3.10.1. Observações Prévias

Antes de apresentarmos uma versão do algoritmo numa forma mais próxima da implementação vamos descrever como os procedimentos empregados por ele devem efetuar algumas operações.

i) No procedimento de Avanço:

Encontrado um nó deficiente e não bloqueado o algoritmo escoa sua deficiência da seguinte forma:

Para cada nó v o algoritmo mantém $\overline{S}(v)$ numa lista ordenada $(r_1, r_2, \ldots, r_{(h)})$ que representaremos por $\mathcal{S}(v)$ e onde $h = |\overline{S}(v)|$. O escoamento de deficiência de v pelos ramos de $\overline{S}(v)$ se dá então, sempre pelo elemento de $\mathcal{S}(v)$ de menor índice que ainda possui alguma folga disponível, ou seja, se usa o arco r_i se $r_1, r_2, \ldots, r_{i-1}$ já tiverem sido saturados anteriormente.

O índice desse primeiro elemento ainda disponível da lista é indicado por um apontador p(v) e é interessante observar que num momento qualquer da execução do algoritmo, entre todos os componentes de $\overline{S}(v)$ somente $r_{p(v)}$ pode não estar nem saturado nem ter fluxo nulo. Isso em particular significa que se:

a) $\mathcal{M}_\ell(v) \equiv$ número de arcos de $\overline{S}(v)$ que tem seu flu alterado pela ℓ-ésima aplicação de Avanço a camada de v;

b) $\gamma_\ell(v) \equiv$ número de arcos de $\overline{S}(v)$ que são saturados nessa aplicação, então será verdade que

$$\mathcal{M}_{\ell}(v) \leq \gamma_{\ell}(v) + 1 \qquad (1)$$

Esse resultado será usado posteriormente na determinação da complexidade do algoritmo.

ii) Em Equilíbrio:

No que diz respeito a rotina de Equilíbrio é interessante ressaltar os dois pontos que indicamos a seguir:

a) O algoritmo guarda para cada nó não bloqueado todos os arcos que trouxeram fluxo até ele desde a última vez em que Equilíbrio foi aplicado a sua camada. São esses arcos que serão utilizados para eliminar a deficiência do nó se a ele próprio viermos a aplicar Equilíbrio. É claro, será suficiente considerarmos apenas esses ramos, pois quando aplicamos equilíbrio ao nó a última vez eliminamos sua deficiência e portanto qualquer novo excesso que tenha se acumulado no nó, só pode ter sido originado por novas chegadas de fluxo a ele a partir desse momento. Além disso, procedendo desse modo evitamos que um nó v da camada V_i ao qual já aplicamos EQ_i volte a se tornar deficiente em execuções posteriores de EQ_{i+1} (2). Isso nos permitirá provar que não se aplica equilíbrio a um nó mais de uma vez.

b) Embora não seja necessário, para não ter de criar outra rotina, resolvemos incluir em EQ_i também a tarefa de verificar se V_{i-1} possui nós deficientes livres ou não. Dessa forma ao final de EQ_i já poderemos optar entre A_{i-1} e EQ_{i-1} conforme haja ou não desses nós em V_{i-1}.

Para que isso seja feito, entretanto, precisamos incluir em Equilíbrio o seguinte procedimento:

"Após eliminar as deficiências de uma camada V_i e aumentando se for o caso, as deficiências de nós de V_{i-1} e declarando os arcos que chegam aos nós de V_i tratados, como bloqueados, para cada $v \in V_{i-1}$ deficiente, faça p(v) indicar o próximo elemento de $\mathcal{S}(v)$ até que $r_{p(v)}$ deixe de ser bloqueado ou que se chegue ao fim da lista quando então se indicará que o próprio nó v está bloqueado".

3.10.2. A Estrutura de Dados

O algoritmo de Karzanov, para obter um fluxo maximal no referente, considera a seguinte estrutura de dados:

1) Para cada camada V_i, uma lista contendo seus nós.
2) Para todo arco $e \in \hat{E}$, um vetor $(a_e, s_e, c(e))$ indicando o nó de entrada, nó de saída e a capacidade do arco, respectivamente.

3) Para cada nó v, uma lista dos seus arcos sucessores: $\overline{S}(v) = (r_1^v, \ldots, r_g^v)$, onde $g = |\overline{S}(v)|$.

4) Para cada nó v, uma lista $\hat{A}(v)$ dos arcos que trazem fluxo até ele desde a última aplicação de equilíbrio a sua camada que inicialmente estará vazia e que representamos por $\hat{A}(v) = (e_1^v, \ldots, e_{|\hat{A}(v)|}^v)$.

Usamos ainda os apontadores p(v) e p'(v) que indicam um elemento de $\overline{S}(v)$ e um de $\hat{A}(v)$, respectivamente.

Além disso usamos as variáveis lógicas "arco-bloqueado (•)" e "Execute-Avanço".

3.10.3. Descrição do Algoritmo

PROGRAMA PRINCIPAL

Para todo $e \in E$, faça:

$\overline{f}(e) = c(e)$

$f(e) = 0$

"arco-bloqueado" (e) = falso

Fim-Para

Para todo $e \in \overline{S}(s)$, faça:

$v = s_e$

$p(v) = 1$

$D(v) = c(e)$

$\hat{A}(v) = \{e\}$

$f(e) = c(e)$

$\overline{f}(e) = 0$

Para todo i de 2 a k, execute:

Para todo $v \in V_i$, faça:

$p(v) = 1$

$D(v) = 0$

$p'(v) = 1$

$\hat{A}(v) = \emptyset$

Fim-Para

Fim-Para

Faça i = 1

Execute Avanço

Fim

AVANÇO

Para cada $v \in V_i$, execute:

Enquanto $D(v) > 0$ e $p(v) \leq |\overline{S}(v)|$, faça:

$p = r^v_{p(v)}$, $q = s_p$

$d = mim\{D(v), \overline{f}(p)\}$

$D(v) = D(v) - d$

$D(q) = D(q) + d$

$f(p) = f(p) + d$

$\overline{f}(p) = \overline{f}(p) - d$

$A(q) = A(q) \cup \{p\}$

$p'(q) = p'(q) + 1$

$p(v) = p(v) + 1$

Fim-Enquanto

Fim-Para

Se i = k-1, Execute Equilíbrio

senão, faça i = i+1 e Execute Avanço

Fim-Avanço

EQUILÍBRIO

Para cada $v \in V_i$, execute:

Se $D(v) > 0$, faça:

$i = 1$,

Repita até $D(v) = 0$

$p' = e_i^v$, $q' = a_{p'}$

$d' = \min \{D(v), f(p')\}$

$D(v) = D(v) - d'$

$D(q') = D(q') + d'$

$f(p') = f(p') - d'$

$\overline{f}(p') = \overline{f}(p') + d'$

"arco-bloqueado" (p') = verdade

$i = i+1$

Fim-Repita

Fim-Se

$p'(v) = 1$

$\hat{A}(v) = \emptyset$

Fim-Para

Se $i = 1$, pare, a solução corrente é um fluxo maximal.

Senão, faça $i = i-1$ e para cada $v \in V_i$, execute:

Se $D(v) > 0$, faça:

Enquanto $p(v) \leq |\overline{S}(v)|$ e "arco-bloqueado" $(r^v_{p(v)})$, faça

$p(v) = p(v) + 1$

Fim-Enquanto

Se $p(v) \leq |\overline{S}(v)|$, "Execute-Avanço" = verdade.

Fim-Se

Se "Execute-Avanço" então execute Avanço, senão execute Equilíbrio

Fim-Equilíbrio

3.11. CONVERGÊNCIA E COMPLEXIDADE DO ALGORITMO

Passamos agora a mostrar que o algoritmo de Karzanov termina em tempo finito e a indicar sua ordem de convergência.

Para isso precisamos dos seguintes resultados:

LEMA K-1

Se um nó é bloqueado pelo algoritmo, ele continuará assim até o final da execução.

Prova

Suponha que isso não seja verdade. Seja então v um nó que o algoritmo bloqueou e posteriormente liberou fazendo voltar a existir um caminho $L = (v = v_0, v_1, v_2, \ldots, v_p = t)$ tal que os arcos (v_i, v_{i+1}), $i = 0, \ldots, p-1$ não estejam saturados. Como antes, quando v estava bloqueado, certamente havia arcos saturados nesse caminho, podemos definir $r_j = (v_j, v_{j+1})$, $j = 0, 1, \ldots, p-1$ como sendo o último dos arcos de L que o algoritmo dessatura antes de v voltar a ser livre. Entretanto a única ocasião em que o algoritmo pode dessaturar esse arco é quando se aplica Equilíbrio v_{j+1} e no caso de v_{j+1} estar bloqueado. Mas nesse caso, mesmo após a dessaturação de r_j, haverá um arco de L $(v_\ell, v_{\ell+1})$, $j+1 \leq \ell \leq p-1$ que estará saturado, contrariando a própria definição de r_j. Desse modo o caminho L não poderá voltar a "liberar" o nó v conforme estamos assumindo.

Portanto a suposição que fizemos é falsa o que demonstra o lema.

C.Q.D.

LEMA K-2

i) Se um nó deixa de ser deficiente, então ele continuará equilibrado até o final do algoritmo.

ii) Toda aplicação de Equilíbrio que precede imediatamente uma de Avanço acaba com a deficiência de pelo menos um nó.

Prova

i) Um nó v deficiente só pode deixar de sê-lo pela aplicação a ele de Equilíbrio e nesse caso v estaria bloqueado nessa ocasião. Como pelo lema K-1, v continuará bloqueado até o fim do algoritmo, Avanço não poderá trazer mais fluxo até ele, assim a única possibilidade desse nó voltar a ser deficiente seria em função de reduzirmos o fluxo num de seus arcos sucessores numa aplicação de equilíbrio a camada seguinte. Tal possibilidade entretanto é eliminada pela própria maneira com que Equilíbrio escolhe os ramos que chegam a um nó para eliminar sua deficiência (2). Desse modo o nó, uma vez equilibrado continuará assim.

ii) A única possibilidade de uma execução de Avanço suceder uma de Equilíbrio é termos que, para algum $i \in \{2,\ldots,k-1\}$, EQ_i ao eliminar a deficiência de um nó v criou outra num nó livre de V_{i-1} quando então executaremos a seguir A_{i-1}. Esse fato é suficiente para demonstrar a afirmação (ii).

C.Q.D.

LEMA K-3

O número de vezes que EQ_j, $j = 1,2,\ldots, k-1$ ou A_i, $i = 1,2,\ldots, k-1$ é executado, está limitado superiormente por $n - 1$ (*).

Prova

Sejam e_j, $j = 1,\ldots, k-1$ e a_i $i = 1,\ldots, k-1$ o número de vezes que EQ_j e A_i são executados, respectivamente.

Observe então que entre duas aplicações sucessivas de EQ_{k-1} temos obrigatoriamente um par EQ_k, A_{j-1}, $j \in \{2,\ldots,k-1\}$ que elimina a deficiência de pelo menos um nó pelo lema K-2. Como ainda por esse lema, um nó que deixou de ser deficiente não pode voltar a sê-lo e como entre esses nós não podem estar a fonte e o poço temos imediatamente que $e_{k-1} \leq (n-2) + 1 = n-1$.

Ora, como para cada execução de EQ_j temos forçosamente uma de EQ_{j-1}, podemos concluir que para todo $j \in \{2,\ldots, k-1\}$, $e_j \leq n - 1$ (3) completando a primeira parte da prova.

Mas também entre duas execuções de A_i temos obrigatóriamente uma de EQ_{k-1} o que significa que para todo $i \in \{2,..,k-1\}$, $a_i \leq e_{k-1} + 1$.

Juntando esse resultado a (3) mostramos que o algoritmo tem convergência finita chegando portanto a atingir as condições de sua regra de parada. Isso em particular significa que após a última execução de EQ_{k-1} não mais se utiliza o procedi

---

(*) Este fato comprova a convergência finita do método.

mento de Avanço e desse modo temos realmente que para todo $i \in \{1,2,\ldots,k-1\}$, $a_i \leq e_{j-1}$ o que junto com (3) encerra a prova.

C.Q.D.

Podemos agora enunciar o resultado principal dessa seção que se refere a complexidade, em tempo, do algoritmo.

TEOREMA K-2

O algoritmo de Karzanov leva um tempo de $O(n^2)$ para obter um fluxo maximal em um referente.

Prova

Vamos limitar, inicialmente, o número de alterações de fluxo nos arcos, efetuadas pelo algoritmo. Só alteramos o fluxo num arco (v,v') durante uma aplicação de Equilíbrio se nessa aplicação eliminamos a deficiência de v'. Como isso, pelo lema K-2 só pode acontecer uma única vez durante a execução do algoritmo, concluimos que também o fluxo de (v,v') só pode ser alterado uma única vez pelo procedimento de Equilíbrio. Considerando então todos os ramos temos portanto que o número total de variações de fluxo neles efetuados por Equilíbrio é limitado por m. (4)

Considere agora:

i) M = número total de atualizações, no valor do fluxo nos ramos, efetuada por Avanço no decorrer de todo o algoritmo.

ii) $M_{i\ell}$ = número de arcos que tem seu fluxo alterado pela ℓ-ésima aplicação de A_i.

iii) $m_{i\ell}$ = número de arcos que são saturados durante a ℓ-ésima execução de A_i.

vi) n_i = número de nós de V_i.

De novo, usando o fato de que um arco, para ter seu fluxo diminuido por Equilíbrio, e em particular ser dessaturado, necessita que seu nó de chegada esteja bloqueado, obtemos que um mesmo arco não será saturado duas ou mais vezes por Avanço e assim:

$$\sum_{i=1}^{k} \sum_{\ell=1}^{a_i} m_{i\ell} \leq m \quad . \tag{5}$$

Além disso, somando todas as desigualdades dadas em (1) temos que:

$$M_{i\ell} \leq m_{i\ell} + n_i$$

e desse modo:

$$M = \sum_{i=1}^{k} \sum_{\ell=1}^{a_i} M_{i\ell} = \sum_{i=1}^{k} \sum_{\ell=1}^{a_i} m_{i\ell} + \sum_{i=1}^{k} \sum_{\ell=1}^{a_i} n_i$$

$$\leq m + (n-1) \sum_{i=1}^{k} n_i = m + (n-1)n \tag{6}$$

a última desigualdade se devendo a (5) e ao Teorema K-1 que afirma ser $a_i \leq (n+1)$, $\forall i \in \{1,\ldots,k\}$.

Para as demais operações e testes efetuados pelo algoritmo, os seguintes limites superiores podem ser estabelecidos:

a) O número de vezes que o algoritmo rotula um ramo como bloqueado em Equilíbrio, é certamente, menor ou igual a m, dado que isso acontece no máximo uma vez para cada ramo (7).

Também o número total de vezes que Equilíbrio, para um dado v altera p(v), pelo fato de $r_{p(v)}$ estar bloqueado é limitado em m por idêntico motivo. (8)

b) O número de vezes que avanço pergunta se um nó é deficiente ou se é bloqueado é com certeza menor do que (n-1).n pois pelo teorema K-1, cada A_i é executado um número de vezes não maior que (n-1). (9)

Juntando então (4), (6), (7), (8), (9) obtemos que o algoritmo bloqueia a fonte num referente em um tempo que não pode ser maior que:

$$O(\max\{m, n(n-1), m+n(n-1)\}) = O(n^2)$$

C.Q.D.

Finalmente observamos que de acordo com o que vimos no capítulo anterior:

a) Não podemos ter mais de n referentes para processar.
b) A construção de cada referente é feita em $O(n^2)$.
c) A atualização dos fluxos após o processamento de um referente é O(m).

Assim, o processamento completo de um referente incluí da sua construção, obtenção de um fluxo maximal de s a t nele e atualização de fluxo nos ramos do grafo original no final do tratamento é $O(n^2)$. Como tais operações não irão ser repetidas mais de n vezes concluímos que o algoritmo de Karzanov determina o fluxo máximo em um tempo não superior a $O(n^3)$.

CAPÍTULO 4

O ALGORITMO MKM

4.1. INTRODUÇÃO

Em 1978, V.M. Malhotra, M.P. Kumar e S.N. Maheshwari |5|, apresentaram o método para determinação do fluxo máximo referenciado na literatura como algoritmo dos "três hindus" ou abreviadamente, conforme o faremos, algoritmo MKM. Esse algoritmo como o de Karzanov tem a complexidade $O(n^3)$, mas é de mais simples formulação.

Como todos os métodos propostos após Dinic, o algoritmo MKM é uma variação desse procedimento em que uma nova tecnologia é aplicada para bloquear a fonte no referente. Afora a mudança nesse procedimento o algoritmo é idêntico ao de Dinic.

4.2. DEFINIÇÕES PRÉVIAS

Para podermos introduzir o procedimento efetuado pelo algoritmo necessitamos de algumas definições prévias que passamos a dar abaixo.

4.2.1. Potencial de Fluxo do Nó

Dados uma rede capacitada G = (V,E,c) dois nós s e t (a fonte e o poço) e um fluxo f de s a t, define-se para cada nó v o que chamaremos de Potencial de Fluxo do Nó, $\rho_f(v)$, da seguinte maneira:

i) se $v \neq s$, $v \neq t$

$$\rho_f(v) = \min \{ \sum_{(v,u) \in \overline{A}(u)} (c(v,u) - f(v,u)), \sum_{(u,v) \in \overline{S}(u)} (c(u,v) - f(u,v)) \} \quad (1)$$

ii) se $v = s$

$$\rho_f(s) = \sum_{(s,v) \in \overline{S}(s)} (c(s,v) - f(s,v)) \quad (2)$$

iii) se $v = t$

$$\rho_f(t) = \sum_{(v,t) \in \overline{A}(t)} (c(v,t) - f(v,t)) \quad (3)$$

O potencial de fluxo de um nó é portanto o mínimo entre a capacidade residual total dos arcos sucessores do nó e a dos arcos antecessores dele, evidentemente, o potencial de fluxo de um nó é a máxima quantidade de acréscimo de fluxo que podemos fazer passar (chegar e sair) pelo nó, assumindo que só existem restrições de capacidade nos arcos adjacentes a ele.

4.2.2. Nó Referencial

Chama-se de Nó Referencial (r_f) a um nó da rede com a propriedade de ter o menor potencial do fluxo, ou seja

$$\rho_f(r_f) = \min_{v \in V} \{\rho_f(v)\} \qquad (4)$$

4.2.3. Potencial Referencial

Se r_f é um nó referencial, chama-se de Potencial Referencial ao potencial de fluxo em r_f.

Quando não houver dúvidas acerca de que fluxo estamos nos referindo, escreveremos r apenas ao invés de r_f.

Definidos os conceitos acima, podemos apresentar o lema no qual se fundamenta o procedimento básico utilizado pelo algoritmo.

LEMA - Seja G = (V,E,c) uma rede em camadas capacitada, s e t dois nós de V e f um fluxo de s a t. Se γ_f é um nó referencial, então é possível obter um fluxo f' cujo valor é o valor de f aumentado em $\rho_f(r_f)$ e para o qual temos $\rho_{f'}(r_f) = 0$.

Prova

Se por acaso, s não for o único nó da primeira camada, então vamos ter um outro nó q nessa camada e portanto com potencial nulo, uma vez que $\overline{A}(q) = \emptyset$. O nó q será então, é claro, um nó referencial e a tese do lema estará trivialmente atendida sem que necessitemos fazer acréscimo de fluxo algum. O

mesmo se dá, caso a última camada não se constitua unicamente de t e portanto podemos assumir, representando a i-ésima camada por V_i, i = 0,1,...,k, que $V_0 = \{s\}$ e $V_k = \{t\}$.

Seja então j a camada de um nó referencial r_f. Vamos mostrar que se introduzimos um acréscimo de fluxo no valor de ρ_f em r_f é sempre possível fazê-lo chegar até o poço. Para verificar isso basta mostrarmos que se esse acréscimo de fluxo valendo ρ_f consegue chegar até a camada V_k, $j \leq k < t$ ele também chegará até V_{k+1} pois então poderemos usar iterativamente esse resultado para provar que é possível fazer com que os δ_f introduzidos cheguem a última camada que contém apenas t.

Sejam então V_k uma camada a qual conseguimos fazer os ρ_f adicionais chegarem e $\delta(n)$, $n \in V_k$, a porção desse acréscimo que atinge nó n. Temos então que:

$$\forall n \quad V_k, \quad \delta(n) \leq \sum_{n \in V_k} \delta(n) = \rho_f \leq \sum_{(n,m) \in \overline{S}(n)} (c(n,m) - f(n,m)) \qquad (5)$$

Assim, os $\delta(n)$ a mais, que chegam a qualquer n de V_k poderão ser escoados pelos ramos que dele saem até a camada V_{k+1} comprovando nossa afirmação.

Invertendo agora os sentidos dos ramos do grafo e é lógico, a ordenação de suas camadas, mostraríamos, procedendo de forma idêntica que igualmente seria possível fazer com que, nessas novas condições, um acréscimo de fluxo de δ_f originado em r_f chegasse até s. Se reinvertermos o sentido dos ramos, isso quer dizer que um aumento de δ_f introduzido em s pode ser levado integralmente até r_f e daí então até t conforme verificamos anteriormente. Pode-se portanto gerar um novo fluxo f' cujo valor é maior que o de f em δ_f e tal que:

$$\delta_{f'}(r_f) = \min \{ \sum_{a \in \overline{A}(r_f)} (c(a) - f(a)) - \delta_f \,,$$

$$\sum_{b \in \overline{S}(r_f)} (c(b) - f(b)) - \delta_f \} = 0 \qquad (6)$$

A última igualdade pela própria definição de δ_f.

C.Q.D.

4.3. O ALGORITMO

Conhecido o lema anterior o procedimento básico empregado pelo algoritmo MKM para bloquear a fonte no referente pode ser induzido. Especificamente esse procedimento é o seguinte:

Considere f um fluxo de s a t no referente R, considere também $\tilde{R} = (\tilde{V},\tilde{E},\tilde{c})$ um subgrafo do referente, dele obtido retirando-se os nós e ramos que em iterações anteriores se determinou estarem bloqueados, e onde $\tilde{c}(e) = c(e) - f(e)$, $\forall e \in \tilde{E}$. No início do algoritmo temos $\tilde{R} = R$ e $f = 0$.

Represente agora por $\tilde{o}$ o fluxo identicamente nulo em $\tilde{R}$, determine então para todo $v \in \tilde{V}$ o potencial $\rho_{\tilde{o}}(v)$ e a seguir um nó referencial $r_{\tilde{o}}$. Obtenha então um fluxo $\tilde{f}$ em $\tilde{R}$ de valor $\rho_{\tilde{o}}(r_{\tilde{o}})$ e tal que $\rho_{\tilde{f}}(r_{\tilde{o}}) = 0$ conforme o lema garante ser possível. Feito isso atualize f fazendo $f(e) = f(e) + \tilde{f}(e)$, para todo $e \in \tilde{E}$ e finalmente obtenha um novo $\tilde{R}$ retirando do atual o nó $r_{\tilde{o}}$, os ramos a ele adjacentes e todos aqueles ramos de $\tilde{R}$ que são saturados por $\tilde{f}$. Após processarmos um $\tilde{R}$ cujo referencial tenha sido s ou t, paramos o algoritmo, em caso contrário repeti

mos o processo.

A regra de parada anteriormente descrita é evidente pois se a fonte for eliminada de $\tilde{R}$, também será de R e portanto não existirá c.a.f. da fonte s ao poço t em R e o fluxo será maximal. O mesmo se pode dizer, é lógico, se eliminamos o poço.

Vamos agora especificar como o algoritmo efetua o transporte em $\tilde{R}$ de um aumento de fluxo de valor $\rho_{\tilde{f}}(\tilde{r})$ da fonte até $\tilde{r}$ e daí até o poço.

O procedimento empregado segue a linha utilizada na demonstração do lema, primeiro introduzindo um acréscimo de fluxo no valor de $\delta_{\tilde{f}}(\tilde{r})$ em $\tilde{r}$ que supomos pertencer a camada $\tilde{k}$ de $\tilde{R}$. Em seguida ele escoa esse acréscimo aumentando o fluxo nos ramos de $\overline{S}(\tilde{r})$ o que introduz deficiências em nós de $\tilde{V}_{\tilde{k}+1}$. Trata de eliminar então as deficiências dessa camada aumentando o fluxo dos ramos que dela saem o que torna deficientes os nós de $\tilde{V}_{\tilde{k}+2}$ e assim sucessivamente até que o poço seja atingido.

Feito isso ele executa idêntico procedimento só que em sentido contrário caminhando na direção da fonte. É evidente que o termo "sentido contrário" é usado não exatamente com o significado gramatical, mas expressando que a "primeira camada" a ser atingida por esse processo será formada pelos nós de $\tilde{V}_{k-1}$ que pertencem a $A(\tilde{r})$, a segunda pelos nós $\tilde{V}_{k-2}$ ligados a esses nós e assim por diante até chegar a $\tilde{V}_0$ ou seja até atingir a fonte s.

Para tornar mais eficiente o processo, a forma empregada pelo algoritmo para escoar a deficiência de um nó tanto quando se anda no sentido do poço como no da fonte é a seguinte:

Inicialmente para cada nó v, colocamos tanto os seus arcos sucessores em R, como os antecessores em listas ordenadas $(s_1,s_2,\ldots,s_q)$ e $(a_1,a_2,\ldots,a_p)$ respectivamente, onde $q = |\overline{S}(v)|$ e $p = |\overline{A}(v)|$. Feito isso, o algoritmo utiliza o seguinte princípio: "um arco (v,s_i), $((a_j,v)$ quando andamos no sentido da fonte), só é usado para escoar uma deficiência temporária em v no tratamento de um dado $\tilde{R}$, se $(v,s_1),\ldots,(v,s_{i-1})$, $((a_1,v),(a_2,v),\ldots,(a_{j-1},v)$ respectivamente) ou não pertencem a $\tilde{R}$ ou acabam de ser saturado no próprio processo de escoamento dessa deficiência". Esse princípio é exatamente o mesmo empregado pelo algoritmo de Karzanov para o escoamento de deficiências no procedimento de Avanço, só que aqui o apresentamos numa outra linguagem.

4.3.1. A Estrutura de Dados

A estrutura de dados com que trabalha o algoritmo que iremos apresentar é semelhante a utilizada pelas versões apresentadas dos algoritmos de Dinic e Karzanov usando o conjunto de nós decomposto em camadas, com os conjuntos $\overline{A}(v)$ e $\overline{S}(v)$ para todo nó e com as trincas (a(e), s(e), c(e)) para todo arco. Como o algoritmo de Karzanov ele utiliza um vetor D(•) de dimensão $|V| = n$ onde guarda as deficiências dos nós e também um apontador $p_1(v)$ para o primeiro arco ainda disponível de $\overline{S}(v)$ que como em Karzanov será uma lista ordenada. A mais apenas o fato de que, como ele "empurra fluxo" em ambos os sentidos a partir do referencial, irá precisar também um indicador $p_2(v)$ para o primeiro elemento disponível de $\overline{A}(v)$ que como vimos em 4.3.

também será ordenado. Além disso para indicar se um ramo permanece no $\tilde{R}$ corrente ou não, ele usa a variável lógica "Não-está-em-$\hat{R}$".

4.3.2. Descrição do Algoritmo

PROGRAMA PRINCIPAL

Dados: k,s,t

(V_i, i = 0,1,...,k)

($\overline{A}(v)$, $\overline{S}(v)$, $\forall v \in V$)

(a(e), s(e), c(e), $\forall e \in E$), execute:

Para todo $e \in E$ faça

$\overline{c}(e) = c(e)$

Não-está-em-$\hat{R}(e)$ = falso

Fim-Para

Para todo i de 1 a k-1, faça:

Para todo $v \in V_i$, execute:

$D(v) = 0$

$p_1(v) = 1$

$p_2(v) = 1$

Fim-Para

Fim-Para

Execute: DETERMINAÇÃO DO PRIMEIRO ρ

Enquanto $\{s,t\} \subseteq \hat{R}$, execute

Tratamento de $\hat{R}$

Se $r \notin \{s,t\}$, execute:

Cálculo de ρ

Fim-Se

Fim-Enquanto

Fim

ROTINA: DETERMINAÇÃO DO PRIMEIRO ρ

Dados $(\overline{S}(v),\ \overline{A}(v),\ \forall v \in V)$ e $(c(e),\ \forall e \in E)$ execute:

Faça $\rho_2(s) = 0$ e

Para cada $e \in \overline{S}(s)$, execute:

$\rho_2(s) = \rho_2(s) + \overline{c}(e)$ (*)

Fim-Para

$\rho = \rho_2(s),\ r = s,\ i_r = 0$

Para cada i entre 1 e k-1, execute:

Para cada $v \in \hat{V}_i$, execute:

$\rho_1(v) = 0$

Para cada $e \in \overline{A}(v)$, execute:

$\rho_1(v) = \rho_1(v) + \overline{c}(e)$

Fim-Para

Se $\rho_1(v) < \rho$ faça $\rho = \rho_1(v),\ r = v,\ i_r = i$

$\rho_2(v) = 0$

Para cada $e \in \overline{S}(v)$, execute:

$\rho_2(v) = \rho_2(v) + \overline{c}(e)$

Fim-Para

Se $\rho_2(v) < \rho$ faça

$\rho = \rho_2(v),\ r = v,\ i_r = i$

Fim-Para

Fim-Para

Faça $\rho_1(t) = 0$ e

Para cada $e \in \overline{A}(t)$, execute:

(*) ρ_1 e ρ_2 são as folgas de "chegada" e "saída" de cada nó respectivamente.

$\rho_1(t) = \rho_1(t) + \bar{c}(e)$

Fim-Para

Se $\rho_1(t) < \rho$, faça $\rho = \rho_1(t)$, $r = t$, $i_r = i$

ROTINA: CÁLCULO DE ρ

Dados $(\rho_1(v), \rho_2(v), \forall v \in V)$, execute:

$\rho = \rho_2(s)$, $r = s$, $i_r = 0$

Para cada i entre 1 e i-1 execute:

Para cada $v \in \hat{V}_i$, execute:

$\bar{\rho} = \text{mim}\ \{\rho_1(v), \rho_2(v)\}$

Se $\bar{\rho} < \rho$ faça $\rho = \bar{\rho}$, $r = v$, $i_r = i$

Fim-Para

Fim-Para

Se $\rho_1(t) \leq \rho$ faça $\rho = \rho_1(t)$, $r = t$, $i_r = k$

Fim-Cálculo ρ

ROTINA: TRATAMENTO DE $\hat{R}$

Faça $D(r) = \rho$, $i = i_r + 1$, $\hat{V}_{i_r} = \hat{V}_{i_r} / \{r\}$

Anda para Frente (r)

Enquanto $i \leq k-1$, execute:

Para cada $v \in \hat{V}_i$ execute:

Anda para Frente (v)

Fim-Para

Faça $i = i-1$

Fim-Enquanto

Faça $D(r) = \rho$, $i = i_r - 1$

Anda para Trás (r)

Enquanto $i \geq 1$ faça

Para cada $v \in V_i$, execute:

Anda para Trás (v)

Fim-Para

Faça i = i-1

Fim-Enquanto

Se $\overline{S}(r) \neq \emptyset$, execute:

Para cada e $\in \overline{S}(r)$ faça

$v' = s(e)$

$\rho_1(v') = \rho_1(v') - \overline{c}(e)$

"Não está em $\hat{R}$" (e) = verdade

Fim-Para

Fim-Se

Se $\overline{A}(r) = \emptyset$, execute

Para cada e $\in \overline{A}(r)$ faça

$v' = a(e)$

$\rho_2(v') = \rho_2(v') - \overline{c}(r)$

"Não está em $\hat{R}$" (e) = verdade

Fim-Para

$V_{i_r} = V_{i_r} / \{r\}$

Fim-Tratamento de $\hat{R}$

ROTINA ANDA PARA FRENTE (v)

Dados ($\{D(v), \rho_1(v), \rho_2(v), p_1(v), p_2(v)\}$, para todo $v \in V$), ($\{f(e), \overline{c}(e)$, "Não está em $\hat{R}(e)$, para todo $e \in E\}$), ($\overline{A}(v)$, $\overline{S}(v)$), execute:

$\rho_2(v) = \rho_2(v) - D(v)$

Enquanto $D(v) > 0$, execute:

Enquanto "Não está em $\hat{R}$"$(e^{v}_{p_1(v)})$ = verdade, faça

$p_1(v) = p_1(v) + 1$

Fim-Enquanto

$e = e^{v}_{p_1(v)}$

$d = mim\ \{\overline{c}(e), D(v)\}$

$D(v) = D(v) - d$

$f(e) = f(e) + d$

$v' = s(e)$

$\rho_1(v') = \rho_1(v') - d$

$D(v') = D(v') + d$

Se $\overline{c}(e) = d$ faça

$p_1(v) = p_1(v) + 1$

"Não está em R"(e) = verdade

Senão, faça

$\overline{c}(e) = \overline{c}(e) - d$

Fim-Enquanto

Fim-Anda para Frente

ROTINA ANDA PARA TRÁS

Dados ($\{D(v), \rho_1(v), \rho_2(v), p_1(v), p_2(v)\}$, $\forall v \in V$),

($\{f(e), \overline{c}(e)$, "Não está em R"(e)$\}$, $\forall e \in V$)

($\{\overline{A}(v), \overline{S}(v)\}$, $\forall v \in V$), execute:

$\rho_1(v) = \rho_1(v) - D(v)$

Enquanto $D(v) > 0$, execute:

Enquanto "Não está em R" $(r^{v}_{p_2(v)})$ = verdade, faça:

$p_2(v) = p_2(v) + 1$

Fim-Enquanto

$e = r^{v}_{p_2(v)}$

$d = mim\ \{\overline{c}(e), D(v)\}$

$D(v) = D(v) - d$

$f(e) = f(e) + d$

$v' = a(e)$

$\rho_2(v') = \rho_2(v') - d$

$D(v') = D(v') + d$

Se $\overline{c}(e) = d$ faça

$p_2(v) = p_2(v) + 1$

Não está em $\hat{R}$" (e) = verdade

Senão, faça:

$\overline{c}(e) = \overline{c}(e) - d$

Fim-Enquanto

Fim-Anda para Trás.

4.4. COMPLEXIDADE DO ALGORITMO

Conforme vimos anteriormente, é sempre possível garantir o seguinte: "Cada vez que fazemos um acréscimo de fluxo de forma a zerar o potencial do nó referencial corrente, no máximo um único arco adjacente a qualquer nó v ainda não eliminado poderá ter seu fluxo modificado sem ser saturado".

Representando então por E_i' o conjunto de arcos que são saturados quando tratamos o i-ésimo $\hat{R}$ ($\hat{R}_i$) podemo afirmar que o número de modificações, no valor do fluxo em arcos, efetuadas nessa aplicação das rotinas Anda para Frente e Anda para Trás a $\hat{R}_i$ é limitada por

$$|V| + |E_i'|$$

onde $|V| = n$.

Somando então em i, verificamos que o número de atualizações de fluxo durante toda a execução do algoritmo tem ordem não superior a

$$O\left(\sum_{i=1}^{|V|} (|V| + |E_i^!|)\right) \leq O(|V|^2 + E) = O(|V|^2) = O(n^2) \qquad (7)$$

Observe ainda que a obtenção dos potenciais pela primeira vez demanda um tempo de

$$O(|E|) \qquad (8)$$

e sua atualização após cada acréscimo de fluxo, que é feita também nas rotinas citadas requer um tempo total de no máximo

$$O\left(\sum_{i=1}^{|V|} |V|\right) = O(|V|^2) = O(n^2) \qquad (9)$$

Além disso para a obtenção de cada referencial resta a operação de determinar o menor elemento não nulo de uma lista de $|V|$ componentes que é $O(|V|)$. Gasta-se portanto com isso, como termos no máximo $|V|$ referenciais, $O(|V|^2)$. (10)

Finalmente, no processo de eliminação dos referenciais dispendemos no máximo $O(|E|)$ em tempo (11). Assim, juntando (7), (8), (9), (10), (11), temos que o algoritmo MKM, bloqueia a fonte no referente em um tempo não superior a $O(|V|^2) = O(n^2)$.

Na obtenção do fluxo maximo, a repetição desse processo para cada referente determinado gastará ao todo $O(|V|^3) = O(n^3)$ operações. Essa será a complexidade, em tempo, do algoritmo, dado que a atualização dos fluxos no grafo original depois de processado um referente e a própria determinação dos referentes não levam um tempo superior a esse conforme apresentamos no capítulo 2.

CAPÍTULO 5

CONCLUSÕES

O Problema de Fluxo Máximo é um dos problemas de Programação Combinatória para os quais se dispõe de uma grande variedade de contruibuições para obter a solução do mesmo.

A primeira delas foi o "antológico" Algoritmo de Ford e Fulkerson o qual é, também, conhecido como o método dos caminhos de acréscimo de fluxo e que procura o fluxo máximo construindo sucessivamente fluxos que cumprem as condições dadas em (1) e (2) da seção 1.2.6. Essa construção do fluxo é executada por meio de acréscimos feitos nos arcos de caminhos que o próprio algoritmo determina.

A procura desses caminhos, entretanto, é feita sem nenhum critério especial para orientá-la e por isso pode acontecer que sejam usados caminhos "longos" o que pode mesmo acarretar, no caso de haver capacidades irracionais, na perda da propriedade de terminação finita do algoritmo.

Depois de um longo período,Edmonds e Karp mostraram que usando sempre caminhos de acréscimo de fluxo de menor tamanho o algoritmo de Ford e Fulkerson se torna garantidamente finito com uma complexidade de tipo polinomial, dependendo só da quantidade de arcos e nós da rede, especificamente $O(nm^2)$.

Segue-se a contribuição mais importante, a de Dinic inventor do referente, que permite a determinação em "bloco" de caminhos de acréscimo de fluxo de tamanho mínimo entre a fonte e o poço. A complexidade melhora então para $O(n^2m)$.

Os três métodos já descritos possuem como características comuns a de utilizar os caminhos para os acréscimo de fluxo e a de trabalhar com fluxos sempre viáveis. Isso entretanto não acontece com o algoritmo de Karzanov o qual faz uso de pseudo "fluxos" denominados prefluxos.

O algoritmo de Karzanov também usa o referente, aquele construido por Dinic, mas a diferença entre ambos os métodos é exatamente a forma de procurar o bloqueio da fonte nesse referente o que significa a obtenção do fluxo maximal. No que se refere a complexidade ele baixa para $O(n^3)$.

O algoritmo MKM a exemplo dos dois anteriores faz uso do referente e só modifica a maneira de obter o fluxo maximal, fazendo uso dos conceitos de potencial e de nó referencial. A complexidade é exatamente a mesma que a de Karzanov.

Além das abordagens descritas acima existem uma série de outras que em sua maioria não trazem propriamente nenhuma metodologia nova para o tratamento do problema, mas apenas conseguem complexidades menores sofisticando a estrutura de dados utilizada. Todos esses métodos que aparecem na lista a seguir junto com os que tratamos nesse trabalho se iniciam com a construção do referente e só modificam o critério para obter nele um fluxo maximal.

Autor	Ano	Complexidade
Ford e Fulkerson	1956	——
Edmonds e Karp	1969	$O(n.m^2)$
Dinic	1970	$O(n^2.m)$
Karzanov	1973	$O(n^3)$
Cherkasky	1977	$O(n^2.m^{1/2})$
Galil	1978	$O(n^{5/3}.m^{2/3})$
Três Hindus	1978	$O(n^3)$
Galil-Naamad	1979	$O(n.m.\log^2 n)$
Shiloach	1979	$O(n.m.\log^2 n)$
Kaplan Sleator	1980	$O(n.m.\log n)$

Entre todos esses novos algoritmos destacamos inicialmente o de Cherkasky que na procura do fluxo maximal divide o referente em super camadas constituidas de várias camadas sucessivas. Dentro de cada super camada, pode-se dizer que o método opera como o de Dinic enquanto que considerando as "supercamadas como camadas", ele trabalha como o de Karzanov. A obtenção dessas supercamadas é feita de tal forma que se obtem um belo exemplo de como misturando dois algoritmos para resolver um mesmo problema, conseguimos um terceiro mais eficiente.

Além desse algoritmo citamos o de Kaplan Sleator dos mais modernos e que usa, para guardar caminhos,uma estrutura de dados chamada "biased 2-3 tree", árvores nos quais as folhas correspondem aos arcos do caminho.

Finalmente gostaríamos de observar que analisando as complexidades da tabela acima, verificamos que elas tendem para o ambicionado O(mn). Entretanto ainda está aberta a questão de se poder garantir que esse limite é, de fato, um "bound" infe

rior para a complexidade em tempo de um algoritmo qualquer de fluxo máximo o que se acredita ser possível. De toda forma, que esse limite, ao qual já estamos chegando, seja válido ou não, a pesquisa relativa ao problema, parece encontrar-se perto da saturação não devendo experimentar nos próximos anos o mesmo desenvolvimento apresentado na última década.

BIBLIOGRAFIA

|[1]| BAZARAA, M.S. and JARVIS, J.J., "Linear Programming and Network Flows", John Wiley and Sons, N.Y., 1977.

|[2]| FORD, L.R. and FULKERSON, D.R., "Flows in Networks", Princeton Univ. Press, Princeton, N.J., (1962).

|[3]| DINIC, E.A., "Algorithm for Solution of a Problem of Maximun Flow in a Network with Power Estimation", Soviet Math. Dokl., 11 (1970), pp. 1277-1280.

|[4]| HU, T.U., "Combinatorial Algorithms", Addison-Wesley Pub. Comp., California, (1982).

|[5]| MALHOTRA, V.M., KUMAR, M.P. e MAESHWARI, S.N., "An $O(V^3)$ Algorithm for Finding Maximum Flows in Networks". Information Processing Letters, Vol. 7, Nº 6, (1978), pp. 277-278.

|[6]| KARZANOV, A.V., "Determining the Max Flow in a Network by the Method of Preflows", Soviet Math. Dokl., Vol. 15, (1974), pp. 434-437.

|[7]| EVEN, Shimon, "The Max Flow Algorithm of Dinic and Karzanov: An Exposition", M.I.T. Lab. for Comp. Sc., Tec. Report, MIT/LCS/TM-80, (1976).

|[8]| GONZAGA, C.C., "Busca em Grafos", I Reunião de Matemática Aplicada, IBM do Brasil, 1978.

|[9]| EDMONDS, J.E. and KARP, R.M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems, J. A.C.M. Vol. 19-(2), (1972), pp. 248-264.

|[10]| FORD, L.R. and FULKERSON, D.R., "Maximal Flow Throuhg a Network", Canadian J.Math., Vol. 8-(3), (1956), pp. 399-404.

|[11]| LAWLER, E.L., "Combinatorial Optimization: Networks and Matroids", Holt Rinehart and Winston, N.Y., (1976).

|[12]| OLIVEIRA, A.A.F. e GONZAGA, C.C., "Convergence Bounds for the "Capacity" Max-Flow Algorithm", monografia do Instituto de Econometria e Pesquisa Operacional da Universidade de Bonn (1982).

|[13]| KAPLAN SLEATOR, DANIEL D., "An O(nm logn) Algorithm for Maximum Network Flow" Tese Ph.D. Stanford University, Stanford, 1980.