"PLANEJAMENTO ÓTIMO DA AMPLIAÇÃO DA CAPACIDADE

DE REDES COM FLUXO"

Maria Tereza Galvão Figueiredo

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 :

Prof. Ronaldo C.Marinho Persiano
(presidente)

Prof. João Lizardo R.H. de Araújo

Prof. Nelson Maculan Filho

Prof. Clovis Caesar Gonzaga

RIO DE JANEIRO - BRASIL
NOVEMBRO DE 1975

AGRADECIMENTOS

- Ao Prof. Ronaldo Marinho pela excelente orientação e pela maneira simples e objetiva como redimiu as minhas dúvidas.

- Aos colegas, professores e funcionários da COPPE que direta ou indiretamente contribuiram para a execução deste traba - lho.

- Ao CNPq pelo auxílio financeiro fornecido durante a obtenção dos créditos e à COPPE pelo auxílio fornecido durante a realização da tese.

RESUMO

O modelo desenvolvido destina-se ao planejamento a curto prazo da ampliação da capacidade de uma rede com fluxo onde podem existir capacidades iniciais.

As ampliações da rede são obtidas por acréscimos de capacidades (inteiros e limitados) aos ramos da rede, os quais importam em um custo. Deseja-se encontrar um número k de ampliações de mais baixo custo da rede, a fim de que uma demanda (não satisfeita na rede inicial) seja atendida.

O problema original é transformado em um problema de busca de k caminhos de menores custos em um grafo, e para a sua resolução apresenta-se três algoritmos: o algoritmo de Programação Dinâmica, uma adaptação do algoritmo de Dijkstra e o algoritmo de Programação Heurística.

ABSTRACT

The model presented is foreseen to the short term planning of the capacity expansion of a flow network, where initial capacities may exist.

The network expansions are obtained by rising the capacity (the values are integers and limited) of the network-arcs, which results in a certain cost. The goal is to find a number k of expansions with lowest cost of the network, so that a demand (not satisfied in the initial network) may be attended.

The original problem is transformed into a problem of the search in a graph of k paths with lowest cost. Three algorithms are presented for the problem resolution: the Dynamic Programming algorithm, an adaptation of the Dijkstra algorithm, and finally the Heuristic Programming algorithm.

INDICE

CAPÍTULO I

I N T R O D U Ç Ã O

O problema aqui estudado surgiu originalmente do problema da ampliação a curto prazo de uma rede de potência. Este trabalho entretanto, limita-se à abordagem teórica do problema.

No problema original considerava-se uma rede de potência constituída de usinas, subestações e centros consumidores (nós da rede) e trechos entre as subestações,onde existem ou podem existir linhas de transmissão (ramos da rede). Por razões técnicas o fluxo de potência que pode circular entre as linhas de transmissão é limitado (capacidade dos ramos).

Devido à limitação do fluxo de potência que circula na rede, o crescimento de mercado e da geração de energia exigem uma ampliação da rede, a fim de que o fluxo de potência atenda as novas necessidades.

Para a ampliação da rede, permite-se a introdução de novas linhas de transmissão ao sistema (acréscimos de capacidades). A introdução destas linhas importam em um custo, e o número de linhas a serem acrescidas em cada trecho é limitado.

Deseja-se determinar o número de linhas a serem introduzidas em cada trecho, a fim de que o fluxo de energia atenda

às exigências do mercado, e o custo de ampliação da rede seja mínimo.

Supõe-se que a capacidade inicial da rede é insuficiente para atender às exigências do mercado.

Observou-se que por razões não só técnicas como também por considerações não incluídas no modelo, a solução de custo mínimo fornece apenas subsídios para a determinação de outras soluções. Assim, formulou-se um segundo problema para determinar outras soluções de baixo custo, a fim de fornecer ao planejador opções na escolha da solução a ser implementada.

Este trabalho tem como objetivo a resolução de um problema como o descrito acima, sem levar em consideração as características da rede de potência. As técnicas utilizadas para a resolução do problema, entretanto, podem ser adaptadas a qualquer tipo de rede de transporte.

No capítulo II formula-se o problema como um problema de programação inteira e justifica-se a não utilização de técnicas de programação inteira para a sua resolução.

No capítulo seguinte reformula-se o problema como um problema de busca de caminhos de baixo custo em um grafo, e apresentam-se dois algoritmos para a sua resolução: o algoritmo de programação dinâmica e um algoritmo baseado no algoritmo de Dijkstra para a busca de um caminho de custo mínimo em um grafo.

No capítulo IV apresenta-se o algoritmo de programação heurística destinado à resolução do problema de busca. No final deste capítulo sugere-se um método para a determinação de uma heurística admissível que garante um funcionamento razoavelmente eficiente deste algoritmo.

Notação

Denota-se por $\mathbb{N}^n$ ($\mathbb{R}^n$) ao conjunto de n-úplas ordenadas de números inteiros (reais) as quais serão consideradas vetores linhas.

Um vetor será caracterizado por uma letra minúscula. No caso de haver mais de um vetor representado pela mesma letra utiliza-se super-índices para diferenciá-los. As componentes de um vetor serão caracterizadas pela letra que representa o vetor e por sub-índices que indicam a componente a que pertencem. Por exemplo, o vetor $x^1 \in R^m$ pode ser representado por $(x_1^1, x_2^1, \ldots, x_m^1)$.

Sejam $x^1, x^2 \in R^m$. Utiliza-se a notação: $x^1 < x^2$ para indicar que $x_i^1 < x_i^2$ para $i=1,2,\ldots,m$. Analogamente $x^1 \leq x^2$ se $x_i^1 \leq x_i^2$, $i=1,2,\ldots,m$. Indica-se $x^1 \leq x^2$ $(x^1 \neq x^2)$ se $x_i^1 \leq x_i^2$, $i=1,2,..,m$, e existe $j=1,2,\ldots,m$ tal que $x_j^1 < x_j^2$.

Um conjunto será caracterizado por uma letra maiúscula e sua cardinalidade indicada pela letra que o caracteriza entre barras. Exemplo: conjunto A, cardinalidade de A: $|A|$. Um conjunto de partes de A será representado por P(A).

Quanto às referências, os capítulos serão enumerados em algarismos romanos, expressões e parágrafos serão indicados por algarismos arábicos. Para referênciar expressão ou parágrafo no mesmo capítulo, utiliza-se sua numeração entre parênteses. No caso de referência a uma expressão ou parágrafo em outro capítulo, indica-se a numeração do capítulo seguida da numeração da expressão

ou parágrafo a ser referenciado. Parágrafos e expressões dos apêndices serão referenciados por letras maiúsculas seguidas do número do parágrafo ou expressão.

A bibliografia é apresentada no final do trabalho em ordem alfabética. A referência à bibliografia é indicada pelo nome do primeiro autor seguido da data de publicação da obra.

CAPÍTULO II

O PROBLEMA

Introdução

Neste capítulo apresenta-se a formulação do problema. A teoria utilizada é a teoria de redes com fluxo - Ver (Ford |62|) e (Berge |62|) ; um resumo desta teoria encontra-se no Apêndice A.

Na secção 1 serão introduzidos os elementos do modelo que será estudado; formula-se um primeiro problema destinado a encontrar uma política de ampliação de custo mínimo e discute-se as razões que levaram à formulação de um segundo problema, destinado a encontrar outras políticas de ampliação além desta; este último problema constitui o objetivo deste trabalho.

Na secção 2 apresenta-se três extensões possíveis do modelo, para os casos: a rede dada possui vários nós de produção e demanda; a rede contém ramos múltiplos, e o caso em que é necessário um aumento de produção.

SECÇÃO 1 - FORMULAÇÃO DO PROBLEMA

Inicia-se esta secção fixando-se os elementos do modelo. Os conceitos básicos de teoria dos grafos que serão utilizados como os conceitos de rede, fluxo em rede, fluxo viável,etc podem ser encontrados no Apêndice A.

Considere-se uma rede orientada $R = (N,M)$ com n nós e m ramos , onde distinguem-se dois nós: a nó fonte e b nó sumidouro.

À rede R estará associado o vetor de capacidades iniciais $c^o \in \mathbb{N}^m$ e o número natural $D \geq 0$ denominado demanda de R .

Diz-se que um fluxo f de a para b em R atende à demanda D se

$$v(f) \geq D$$

Nos casos de interesse a rede R não admite nenhum fluxo viável em relação a c^o , que atenda à demanda D . Admite-se portanto ampliações da capacidade dos ramos através de acréscimos de capacidade: para cada ramo r_i é dado um acréscimo de capacidade de valor $\Delta c_i \in \mathbb{N}$;permite-se que a capacidade do ramo r_i seja ampliada através de um número inteiro e limitado de acréscimos Δc_i . O vetor $\Delta c \in \mathbb{N}^m$ representa os acréscimos permissíveis.

No problema aqui estudado busca-se políticas de ampliações das capacidades iniciais de forma a atender à demanda e o

timizar custos que serão definidos mais adiante.

A uma possível política de ampliação (que atenda ou não à demanda) denomina-se configuração.

1 Definição : Uma configuração C é uma tripla ordenada (N,M,s) onde N e M são os elementos de R e $s \in \mathbb{N}^m$ é um vetor de estado.

O vetor de estado s de uma configuração caracteriza perfeitamente a política de ampliação: s_i é o número de acréscimos de valor Δc_i feitos ao ramo r_i. O estado $s^o=0$ representa a configuração inicial, isto é : com capacidade igual à capacidade inicial.

A capacidade do ramo r_i na configuração (N,M,s) é dada por :

$$2 \qquad c_i(s) = c_i^o + s_i \Delta c_i$$

O vetor $c(s) = (c_1(s), c_2(s), \ldots, c_m(s))$ é o vetor capacidade da configuração C.

Devido a (2), s determina completamente a configuração C, uma vez que (N,M) é fixada. Daqui por diante, por razões de comodidade, utiliza-se a notação s em lugar de $C=(N,M,s)$ identificando-se o estado com a configuração.

Um fluxo f é dito admissível em s se

a) f atende a demanda

b) $f(r_i) \leq c_i(s) \quad , \quad \forall r_i \in M$

Denota-se por $t \in \mathbb{N}^m$ o <u>vetor de limite de acréscimos permissíveis</u> onde t_i é o maior número de acréscimos admitidos no ramo r_i .

Uma configuração s é dita <u>configuração viável</u> se:

a) $0 \leq s \leq t$

b) existe um fluxo admissível em s .

Uma configuração viável é portanto uma ampliação que permite atender-se à demanda com fluxos viáveis.

A ampliação da configuração inicial importa em um custo, que será a soma dos custos de ampliação de cada ramo. Portanto, o <u>custo de uma configuração s</u> é dado por :

$$p(s) = \sum_{i=1}^{m} p_i(s_i)$$

onde $p_i : \mathbb{N} \longrightarrow R^+$ é a função custo do ramo r_i. $p_i(s_i)$ é o custo da introdução de s_i acréscimos de valor Δc_i ao ramo r_i. Supõe-se que as funções p_i são estritamente crescentes, $p_i(s_i)$ é finito $\forall s \leq t$ e $p_i(0) = 0$.

Formulação do Problema

Com os elementos expostos pode-se formular um primeiro problema.

(P1) Encontrar, se existir, uma configuração viável de menor custo entre todas as configurações viáveis.

O problema (P1) admite solução desde que exista uma configuração viável, pois o conjunto de configurações s satisfazendo $s \leq t$ é finito. Supõe-se que a configuração s=t é viável.

Métodos de Resolução do Problema (P1)

No caso particular em que as funções custos são lineares e os acréscimos de capacidade unitários, o problema (P1) pode ser transformado num problema de circulação e resolvido aplicando-se out-of-kilter como exposto em Ford |62|.

Pode-se ainda resolvê-lo através de técnicas de programação inteira; a sua formulação neste caso resultaria no problema dado a seguir, com 2m variáveis e 2m+n-1 restrições. (Para a resolução deste problema, ver Hu |69|.

Dados: $R = (N,M)$, $a \in N$, $b \in N$

$D \in \mathbb{N}$

$c^o \in \mathbb{N}^m$

$\Delta c \in \mathbb{N}^m$

$t \in \mathbb{N}^m$

$$p_i : \mathbb{N} \longrightarrow \mathbb{R}^+ \quad , \quad r_i \in M$$

3 Minimizar $\sum_{i=1}^{m} p_i(s_i)$

sujeito a

$$f(n,N) - f(N,n) = 0 \qquad \forall\, n \in N \;,\; n \neq a,b$$

$$f(N,b) - f(b,N) \geq D$$

4 $$0 \leq f(r_i) \leq c_i^o + s_i \Delta c_i \qquad \forall r_i \in M$$

$$0 \leq s_i \leq t_i \qquad \forall r_i \in M$$

$$s_i \text{ inteiro}$$

Quando se trata de aplicações do modelo, uma solução do problema (P1) pode não ser muito significativa, pois, devido a considerações não incluídas no modelo, ela poderia ser rejeitada como uma possível decisão implementável. Procura-se então determinar um número $k \in \mathbb{N}$ de configurações viáveis de mais baixo custo, a fim de se ter um maior número de alternativas a serem escolhidas.

A determinação das k configurações de mais baixo custo poderia ainda fornecer soluções inadequadas pois entre elas poderiam surgir configurações pouco eficientes. Por exemplo, a segunda configuração de menor custo poderia conter acréscimos dispensáveis porém de baixos custos; ou seja, esta configuração poderia ser obtida a partir da primeira de menor custo através de al -

guns acréscimos em ramos de custos muito baixos.

O argumento acima sugere a introdução da seguinte definição.

5 Definição: Uma configuração s é dita viável não supérflua se :

6 $(\nexists\ s' \leq s\ ,\ s' \neq s)$ s' é não viável

A definição (5) explicita que uma configuração é viável não supérflua se todos os acréscimos de capacidade feitos para obtê-la são indispensáveis , ou seja, a retirada de qualquer destes acréscimos alterariam sua viabilidade.

Uma configuração é dita viável supérflua se é viável e não satisfaz a condição (6).

O problema que se propõe solucionar no presente trabalho é enunciado a seguir.

(Pk) Encontrar k configurações viáveis não supérfluas (ou todas, caso não existam k) de mais baixo custo entre todas as configurações viáveis não supérfluas.

No capítulo seguinte demonstra-se que se R admite uma configuração viável, então R admite uma configuração viável não supérflua; como se supõe que a configuração s=t é viável , pode-se garantir que o problema (Pk) tem sempre solução.

O problema (Pk) é equivalente ao problema (P1) quando k=1. Nos demais casos a resolução do problema (Pk) através de

técnicas de programação inteira seria muito difícil devido à complexidade das restrições e ao número de alternativas exigido. Note-se que verificar se uma configuração viável s é ou não supérflua resultaria num processo bastante dispendioso, uma vez que dever-se-ia testar a viabilidade de cada uma das configurações obtidas a partir de s , pela supressão de um acréscimo presente em s .

Nos capítulos III e IV serão apresentados métodos de resolução para o problema (Pk) baseados em técnicas de proble - mas de busca de caminho mínimo em grafos.

Na secção seguinte apresenta-se algumas extensões do modelo.

SECÇÃO 2 - GENERALIDADES DO MODELO

O modelo tratado neste trabalho é suficientemente geral para englobar uma série de problemas particulares que podem ocorrer na prática. Nesta secção apresenta-se alguns destes problemas e como manipulá-los a fim de transformá-los no modelo aqui estudado.

7 Redes com Ramos Múltiplos

Dada a rede R=(N,M) onde está definido c^o vetor capacidade inicial da rede e D demanda de R , considere-se o ca

so em que a um ramo r_i qualquer de R estão associados vários possíveis acréscimos de capacidade de valores Δc_i^j , $j=1,2,\ldots,q_i$. A cada acréscimo Δc_i^j está associado t_i^j , número máximo de acréscimos Δc_i^j permissíveis ao ramo r_i e $p_i^j(.)$ função custo associada ao ramo r_i .

Deseja-se encontrar números s_i^j de acréscimos de capacidade Δc_i^j , $j=1,\ldots,q_i$, $r_i \in M$ tal que a demanda da rede seja satisfeita e o custo de ampliação seja mínimo (a transformação que será feita neste problema permite a determinação não só da solução de custo mínimo como das k soluções de menores custos).

A formulação deste problema como problema de programação inteira é dada a seguir.

Dados: $R = (N,M)$, $a \in N$, $b \in N$

$D \in \mathbb{N}$

$c^o \in \mathbb{N}^m$

$\Delta c_i^j \in \mathbb{N}$, $j=1,\ldots,q_i$, $r_i \in M$

$p_i^j : \mathbb{N} \rightarrow \mathbb{R}$, $j=1,\ldots,q_i$, $r_i \in M$

8 Minimizar $\sum_{j=1}^{q_i} \sum_{i=1}^{m} p_i^j(s_i^j)$

sujeito a

$f(n,N) - f(N,n) = 0 \qquad \forall n \in N , n \neq a,b$

$f(N,b) - f(b,N) \geq D$

9
$$0 \leq f(r_i) \leq c_i^o + \sum_{j=1}^{q_i} s_i^j \, \Delta c_i^j \qquad r_i \in M$$

$$0 \leq s_i^j \leq t_i^j \qquad r_i \in M \, , \; j=1,2,\ldots,q_i$$

$$s_i^j \text{ inteiro} \qquad r_i \in M \, , \; j=1,2,\ldots,q_i$$

Nota-se que a função objetiva e o conjunto de restrições do problema (8) diferem do problema (3) devido ao fato de aqui ter-se mais de um acréscimo de capacidade associados a um mesmo ramo da rede.

A transformação do problema acima no problema estudado será ilustrada por um exemplo.

Exemplo I : Considere-se a rede $R=(N,M)$ onde

$N = \{n_1, n_2\}$

$M = \{r\}$, como mostra a figura 1 .

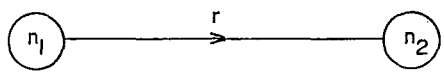


figura 1

O ramo r tem como capacidade inicial c^o e a ele estão associados três valores de acréscimos de capacidade Δc^1 , Δc^2, Δc^3. O número de acréscimos Δc^j em r é limitado por t^j , $j=1,2,3$. Ao ramo r estão associadas as funções custo $p^j(.)$ $j=1,2,3$.

Como o problema original admite um único tipo de acréscimo de capacidade para cada ramo r_i, criam-se q ramos, e a cada ramo r_i^j , j=1,...,q associa-se Δc_i^j , t_i^j , $p_i^j(.)$.

Faz-se uma distribuição de capacidade inicial nos ramos r_i^j tal que

$$\sum_{j=1}^{q} c_i^{oj} = c_i^o \qquad r_i \in M \ , \ c_i^{oj} \in \mathbb{N} \ , \ c_i^o \in \mathbb{N}$$

ou seja, a soma das capacidades iniciais dos ramos criados deve ser igual à capacidade inicial dada.

A rede do exemplo I pode ser transformada na rede da figura 2.

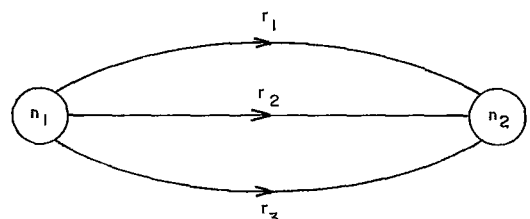


figura 2

onde: ao ramo r_1 está associado: $c^{o1} = c^o$, Δc^1, t^1, $p^1(.)$

ao ramo r_2 está associado: $c^{02} = 0$, Δc^2, t^2, $p^2(.)$

ao ramo r_3 está associado: $c^{03} = 0$, Δc^3 , t^3, $p^3(.)$

A solução do problema fornece os s^j , j=1,2,3 , significando que se deve fazer s^j acréscimos do tipo Δc^j ao ramo r para se obter uma configuração viável de R de menor custo.

A capacidade do ramo r na rede original será dada por :

$$c^o + \sum_{j=1}^{3} s^j \Delta c^j$$

Observa-se que esta transformação é possível uma vez que a rede do modelo admite ramos múltiplos (Ver Apêndice A).

10 Vários Nós de Produção e Demanda

Considere-se a rede $R = (N,M)$ onde estão definidos: c^o, t, $\Delta c \in \mathbb{N}^m$ e as funções $p_i(.)$ como no problema original. Aqui, porém, o conjunto de nós de R, N, está particionado em três conjuntos.

A - conjunto de nós produtores

I - conjunto de nós intermediários

B - conjunto de nós consumidores

A cada elemento $a_i \in A$ está associado um número natural ℓ_i denominado capacidade de produção do nó a_i.

A cada elemento $b_i \in B$ está associado um número natural d_i denominado demanda do nó b_i.

Define-se um fluxo admissível de A para B como sendo uma função $f : M \longrightarrow \mathbb{R}$ tal que

$$f(a_i,N) - f(N,a_i) \leq \ell_i \qquad \forall\, a_i \in A$$

$$f(n,N) - f(N,n) = 0 \qquad \forall\, n \in I$$

$$f(N,b_i) - f(b_i,N) \geq d_i \qquad \forall b_i \in B$$

$$0 \leq f(r_i) \leq c_i \qquad \forall r_i \in M$$

onde c_i é a capacidade do ramo r_i .

Deseja-se encontrar s_i , número de acréscimos de capacidade do ramo $r_i \in M$ tal que a nova configuração $s = (s_1, s_2,\ldots,s_m)$ comporte um fluxo admissível f de A para B e o custo de s seja mínimo. (A transformação que será feita permite a determinação das k configurações de menores custos que comportam um fluxo admissível f).

A formulação deste problema como problema de programação inteira é dada a seguir.

Dados:

$R = (N,M)$; $N = A \cup I \cup B$

$d \in \mathbb{N}^{|B|}$

$\ell \in \mathbb{N}^{|A|}$

c^o, Δc , $t \in \mathbb{N}^m$

$p_i : \mathbb{N} \longrightarrow \mathbb{R}$, $r_i \in M$

11 $$\text{Min} \sum_{i=1}^{m} p_i(s_i)$$

sujeito a

$$f(n,N) - f(N,n) = 0 \qquad \forall n \in I \ , \ n \neq a,b$$

12 $$f(N,b_i) - f(b_i,N) \geq d_i \qquad \forall b_i \in B$$

13 $$f(a_i,N) - f(N,a_i) \leq \ell_i \qquad \forall a_i \in A$$

$$0 \leq f(r_i) \leq c_i^o + s_i \Delta c_i \qquad \forall r_i \in M$$

$$0 \leq s_i \leq t_i \qquad \forall r_i \in M$$

$$s_i \text{ inteiro}$$

14 Pode-se mostrar que este problema só tem solução viável se a soma das capacidades de produção dos nós a_i for maior ou igual à soma das demandas dos nós b_i , ou seja

$$\sum_{i=1}^{|A|} \ell_i \geq \sum_{i=1}^{|B|} d_i$$

O exemplo II ilustra um dos processos de transformação deste problema no problema original, estudado neste trabalho.

Exemplo II : Seja $R = (N,M)$ onde

$N = A \cup I \cup B$

$A = \{a_1,a_2\}$

$B = \{b_1,b_2\}$

I é o conjunto de nós limitados pelo retângulo da figura 3 que representa a rede R .

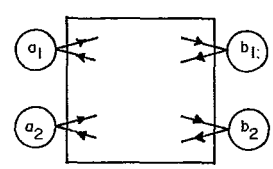


figura 3

Na rede R estão definidos os dados do problema (11) onde ℓ_i é a capacidade de produção do nó $a_i \in A$ e d_i é a demanda do nó $b_i \in B$.

Para a transformação cria-se dois nós fictícios a e b que serão ligados à rede R pelos ramos (a, a_i), $a_i \in A$; e $(b_i, b) \in B$.

Pela restrição (13) ℓ_i é a maior quantidade de fluxo que pode sair do nó a_i ; logo a capacidade do ramo (a, a_i) será feita igual a ℓ_i .

A restrição (12) exige que a quantidade de fluxo que chega a b_i seja pelo menos d_i , assim a capacidade do ramo (b_i, b) será feita igual a d_i , e esta capacidade deverá estar esgotada nas soluções viáveis.

A figura 4 mostra a rede do exemplo II transformada.

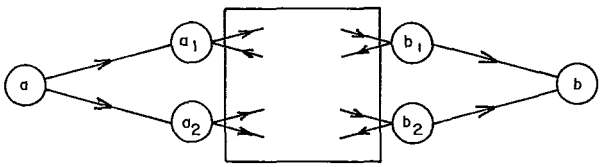


figura 4

Aos ramos (a, a_i) , $a_i \in A$ e (b_i, b), $b_i \in B$ associam-se acréscimos de capacidade, limites de acréscimos permissíveis e funções custos nulos , uma vez que não teria sentido fazer-

se acréscimos de capacidade e atribuir-se custos aos ramos fictí - cios.

A demanda da rede é feita

$$D = \sum_{i=1}^{|T|} d_i$$

Suponha-se que foi encontrada a solução do proble- ma utilizando-se a rede $R'= (N',M')$ da figura 4, onde a demanda da rede é dada por

$$D = d_1 + d_2$$

e os acréscimos de capacidade, os limites de acréscimos permissí - veis e os custos associados aos ramos (a,a_i) e (b_i,b) , $i=1,2$ são todos nulos. Esta solução fornecerá uma configuração viável de R' de custo mínimo. Logo existe um fluxo f admissível em s, ou seja

15 $$v(f) \geq d_1+d_2$$

16 $$f(r_i) \leq c_i(s) \quad \forall r_i \in M'$$

Como os acréscimos de capacidade nos ramos (b_i,b) , $i=1,2$ são nulos, as capacidades destes ramos na configuração s são respectivamente d_1 e d_2; logo, devido a (15) e (16)

$$v(f) = d_1 + d_2$$

ou seja, o fluxo que chega aos nós b_1 e b_2 são respectivamente d_1 e d_2 e portanto este fluxo é admissível em R .

Como não são permitidos acréscimos de capacidade aos ramos (a,a_i) e (b_i,b) , $i=1,2$, então os acréscimos de capacidade feitos para a obtenção da configuração s ocorreram apenas nos ramos da rede original R da figura 3, donde se conclui que s é uma configuração viável desta rede, de custo mínimo.

17 Planejamento de Acréscimo de Produção

Considere-se a rede R do exemplo II e suponha-se que:

$$d_1 + d_2 > \ell_1 + \ell_2$$

como observou-se em (14) o problema não tem solução viável.

Considera-se aqui o problema de ampliar não só as capacidades da rede como as capacidades de produção através dos acréscimos de produção de valor $\Delta\ell_i$ associado a cada $a_i \in A$. O número de acréscimos de produção no nó $a_i \in A$ está limitado por $\bar{t} \in \mathbb{N}$ de tal forma que :

$$\sum_{i=1}^{|A|} \ell_i + \bar{t}_i \Delta\ell_i \geq \sum_{i=1}^{|B|} d_i \qquad 18$$

A expressão (18) obriga que o número de acréscimos de produção permitidos aos nós a_i sejam tais que a soma das produções acrescidas seja maior ou igual a demanda da rede; se esta condição não for satisfeita o problema continua sem solução.

A cada nó $a_i \in A$ estará associada uma função $\bar{p}_i(.)$ de aumento de produção.

Deseja-se encontrar s_j , número de acréscimos de capacidade no ramo $r_j \in M$ e $\bar{s}_i$, número de acréscimos de produção no nó $a_i \in A$, tal que a demanda da rede dada por $\sum_{i=1}^{|B|} d_i$ seja satisfeita e o custo de ampliação seja mínimo.

Formulando este problema como problema de programação inteira tem-se :

Dados:

$R = (N,M)$, $N=A \cup I \cup B$

c^o, t, $\Delta c \in \mathbb{N}^m$

$p_j(.) : \mathbb{N} \longrightarrow \mathbb{R}$

$d \in \mathbb{N}^{|B|}$

$\ell \in \mathbb{N}^{|A|}$

$\bar{t}$, $\Delta\ell \in \mathbb{N}^{|A|}$

$\bar{p}_i(.) : \mathbb{N} \longrightarrow \mathbb{R}$

Minimizar $\sum_{j=1}^{m} p_j(s_j) + \sum_{i=1}^{|A|} \bar{p}_i(\bar{s}_i)$

sujeito a :

$f(n,N) - f(N,n) = 0 \quad \forall n \in I$

$f(N,b_i) - f(b_i,N) \geq d_i \quad \forall b_i \in B$

$f(N,a_i) - f(a_i,N) \leq \ell_i + \bar{s}_i \Delta\ell_i \quad \forall a_i \in A$

$$0 \leq f(r_j) \leq c_j^o + s_j \Delta c_j \qquad \forall r_i \in M$$

$$0 \leq s_j \leq t_j \qquad \forall r_j \in M$$

$$0 \leq \overline{s_i} \leq \overline{t_i} \qquad \forall a_i \in A$$

$$s_j \text{ e } \overline{s_i} \text{ inteiros}$$

A transformação deste problema no problema do modelo tratado é feita de maneira análoga à transformação de (10), sendo que aos ramos (a,a_i) , $a_i \in A$ estão associados além das capacidades de produção ℓ_i , os acréscimos de produção $\Delta\ell_i$, os limites de acréscimos $\bar{t}_i$ e as funções custos $\bar{p}_i(.)$.

Conclusões

Neste capítulo apresentou-se a formulação do problema (P1) como um problema de programação inteira. A resolução deste problema por meio desta técnica é em geral muito trabalhosa devido ao número de restrições a que ele está vinculado. Observou-se também da dificuldade da resolução do problema (Pk) através de técnicas de programação inteira, não só pela complexidade das restrições, como também pelo número de alternativas exigidas.

Técnicas de programação dinâmica poderiam ser aplicadas para a resolução do problema (Pk); porém o custo operacional seria muito alto, o que invalida a utilização do processo.

No capítulo seguinte estuda-se a reformulação de (Pk)

como um problema de busca em grafos, e apresentam-se dois algoritmos para a busca: o primeiro baseado em técnicas de programação dinâmica, e o segundo utilizando-se resultados da teoria desenvolvida por Dijkstra para a busca de caminhos de custo mínimo em um grafo.

CAPÍTULO III

REFORMULAÇÃO DO PROBLEMA

Introdução

Neste capítulo apresenta-se a reformulação de (Pk) como um problema de busca de caminhos em grafos.

Na secção 1 encontram-se os elementos que possibilitam a determinação do grafo de configurações onde será efetuada a busca. O primeiro grafo provém de uma idéia intuitiva na determinação de sucessores. Este grafo fornece uma boa informação da estrutura do grafo que se está buscando; entretanto, será posto de lado devido a sua dimensão. Um segundo grafo de configurações baseado nas propriedades estruturais da rede e das configurações viáveis é apresentado em seguida.

Na secção 2 encontram-se os algoritmos destinados à busca dos k caminhos de menores custos no grafo.

Secção 1 - O GRAFO DE BUSCA

Com o problema (Pk) deseja-se determinar k configurações viáveis não supérfluas de menores custos, obtidas através de acréscimos de capacidade aos ramos da rede R , a partir da configuração inicial $C^0 = (N,M,0)$.

Este problema pode ser reformulado como um problema de busca de caminhos em um grafo $G = (T,\Gamma)$ onde T é o conjunto de nós de G dado por

1 $$T = \{s \in \mathbb{N}^m \;/\; s \leq t\}$$

e $\Gamma : T \longrightarrow P(T)$ é o operador sucessor que associa a cada nó de T um subconjunto de nós de T (Ver B1).

A primeira idéia para a determinação do operador sucessor Γ seria tentar-se, a partir da configuração inicial, um acréscimo de capacidade em cada ramo da rede indistintamente, obtendo-se após cada acréscimo uma configuração maior que s^0 em uma componente. Procedendo-se de maneira análoga com as novas configurações até que não seja mais possível nenhum acréscimo, ou seja, até a configuração $s=t$ ser gerada. Encontra-se, dessa forma, um grafo de configurações onde o número de sucessores de cada configuração é próximo de m (número de ramos de R), uma vez que para obter-se as sucessoras de uma configurações tentou-se acréscimos em todos os ramos da rede. Este grafo contém todas as configurações de R, inclusive as viáveis.

O grafo G obtido através deste operador sucessor seria da forma $G = (T,\Gamma)$ onde T é definido por (1) e $\Gamma:T \to P(T)$ é dado por

2 $$\Gamma(s) = \{s' \in \mathbb{N}^m \;/\; s' \leq t \;,\; s'=s+e_i \;,\; r_i \in R\}$$

Observa-se que se $s' \in \Gamma(s)$ então s' é obtida a partir de s por um acréscimo de capacidade em um ramo de R ;

os ramos do grafo G são da forma (s,s'), $s' \in \Gamma(s)$.

O <u>custo do ramo</u> (s^1,s^2) de G é dado por

3 $$\hat{p}(s^1,s^2) = p(s^2) - p(s^1)$$

O custo do caminho $s^0, s^1,\ldots,s^j$ (Ver B4) em G é portanto igual a $p(s^j)$ uma vez que

4 $$\sum_{i=0}^{j-1} \hat{p}(s^i,s^{i+1}) = \sum_{i=0}^{j-1} p(s^{i+1}) - p(s^i)$$

$$= p(s^j) - p(s^0)$$

$$= p(s^j)$$

Em consequência de (4), a configuração terminal de um caminho de custo mínimo de s^0 a uma configuração viável, é uma configuração viável de menor custo. Logo se o grafo contiver caminhos de s^0 a todas as configurações viáveis não supérfluas, então as k configurações de menores custos obtidas através da busca dos k caminhos de menores custos ligando s^0 ao conjunto de configurações viáveis não supérfluas constitui uma solução do problema (Pk) .

Mostra-se que o grafo G definido por (1) e (2) contêm caminhos de s^0 a todas as configurações viáveis. De fato, considere-se uma configuração viável qualquer s , e retire-se de s uma unidade de cada vez de cada uma das suas componentes positivas; obtem-se após cada retirada uma configuração $0 \leq s^j \leq s$. Continuando-se o processo até encontrar a configuração $s^0=0$, gerou -

se em G um caminho de s^0 a s .

Portanto, através do grafo G definido por (1) e (2), pode-se encontrar a solução do problema (Pk). Este grafo, porém, contém um número muito grande de caminhos pouco promissores , uma vez que são feitos acréscimos em todos os ramos de R indistintamente para a obtenção das sucessoras de uma configuração.Consequentemente a busca neste grafo seria muito trabalhosa. Procurou-se então encontrar um grafo que contivesse menos caminhos através do qual fosse possível a resolução do problema (Pk).

A seguir introduz-se alguns conceitos necessários à determinação do novo grafo.

Dado um corte K da rede R (Ver A8), a capacidade deste corte na configuração s é dada por :

$$c(K,s) = \sum_{i \in (X,\bar{X})} c_i(s)$$

onde $c_i(s)$ foi definido em (II2).

Diz-se que um corte é mínimo na configuração s , se sua capacidade é mínima entre as capacidades de todos os cortes em s .

Propriedade das configurações viáveis

Na proposição seguinte utiliza-se a solução do problema de fluxo máximo (Ver (A10), na caracterização de configurações viáveis.

6 Proposição : Sejam uma configuração $0 \leq s \leq t$ e um fluxo máximo $\hat{f}$ em s ; então, s é viável se e somente se

$$v(\hat{f}) \geq D$$

Demonstração: Se s é viável, então $0 \leq s \leq t$ e existe um fluxo f, admissível em s. Se $\hat{f}$ é fluxo máximo de s, então:

$$v(\hat{f}) \geq v(f)$$
$$\geq D \quad \text{pois} \quad t \text{ é admissível}$$

A implicação inversa é consequência direta da definição de configurações viáveis. ||

O corolário seguinte é uma decorrência imediata do teorema de fluxo máximo - corte mínimo (A11), e da proposição (6).

7 Corolário : Uma configuração s é viável se e somente se $0 \leq s \leq t$ e a capacidade de qualquer corte mínimo na configuração s é maior ou igual a D.

O Grafo G

O corolário (7) fornece a base para a determinação do operador sucessor do novo grafo; o processo utilizado é dado a seguir.

Seja s uma configuração não viável. Pelo corolário (7), existe um corte mínimo em s cuja capacidade é menor que D. Logo, para obter-se a viabilização de s, torna-se necessário pelo menos um acréscimo de capacidade em algum ramo deste cor

te. Observa-se que a configuração resultante de um acréscimo desse tipo comporta um fluxo de valor maior ou igual ao fluxo máximo de s .

O operador sucessor que será definido faz um acréscimo de capacidade em cada ramo de um dado corte mínimo de s , obtendo assim o conjunto de configurações sucessoras de s . Aumenta-se desta forma a capacidade dos cortes mínimos das configurações, até que isto não seja mais possível; ou seja, até encontrar as configurações cujos ramos dos cortes mínimos tenham alcançado o limite de acréscimos.

Como podem existir vários cortes mínimos em uma configuração, define-se a seguir uma aplicação H que estipula o corte mínimo a ser utilizado pelo operador sucessor do grafo G .

Seja $\mathcal{H}$ o conjunto de todos os cortes de R e T o conjunto de todas as configurações de R . Define-se $H : T \to \mathcal{H}$ como sendo uma aplicação que a cada $s \in T$ associa um determinado corte mínimo de s . Em (15) sugere-se um procedimento que caracteriza um corte mínimo de cada configuração s .

O novo operador sucessor $\Gamma : T \to P(T)$ é dado por:

$$\Gamma(s) = \{s' \in \mathbb{N}^m \;/\; s' \leq t \;,\; s' = s + e_i \;,\; r_i \in H(s)\} \qquad (8)$$

Observa-se que $s' \in \Gamma(s)$ se e somente se existe $i = 1,\ldots,m$, tal que

a) $s'_j = s_j \quad j=1,\ldots,m \;,\; j \neq i$

b) $s'_i = s'_i + 1 \leq t_i \quad , \quad r_i \in H(s)$

O grafo $G = (T,\Gamma)$, que será utilizado pela busca fica estabelecido por (1) e (8). A busca neste grafo torna-se mais eficiente que no grafo definido de início, por conter em geral um número menor de ramos (os acréscimos de capacidade são feitos exclusivamente em ramos da rede pertencentes a cortes mínimos das configurações) e consequentemente um número menor de caminhos.

O custo do ramo (s^1,s^2) , $s^2 \in \Gamma(s^1)$ é dado em (3).

De (3) resulta que o custo de um caminho ligando s^0 a s no grafo G é igual ao custo de s . Note-se que dois caminhos distintos ligando s^0 a s correspondem a um mesmo esquema de ampliação, diferindo apenas na ordem em que são introduzidos os acréscimos. Portanto, possuem custos iguais, o que pode ser verificado através de (3) e (4).

9 Definição : Denomina-se alvo ao conjunto T^* de todas as configurações viáveis não supérfluas de R .

Uma solução de (Pk) consiste portanto em um conjunto de configurações s^i , $i=1,\ldots,k$, pertencentes a T^* e de menores custos entre todas as configurações de T^*.

Problema de busca de caminhos de baixo custo em G

Pbk Encontrar k caminhos em G de s^0 a configurações distintas de T* (ou todas, caso não existam k) de mais baixo custo entre todos os caminhos de s^0 a T* .

Observa-se que as configurações terminais dos caminhos obtidos pela resolução de (Pbk) é uma solução de (Pk), desde que existam caminhos em G ligando s^0 a todas as configurações de T*. A propriedade que será apresentada adiante garante a obtenção de uma solução de (Pk) através da resolução de (Pbk).

10 Lema : Seja s uma configuração viável; então existe uma configuração $\bar{s} \leq s$ tal que $\bar{s}$ é viável não supérflua.

Demonstração : Seja s uma configuração viável.

Seja $X = \{s' \leq s \;/\; s' \text{ é viável}\}$

Seja $\bar{s} \in X$ tal que

$$p(\bar{s}) \leq p(s') \quad \forall s' \in X$$

$\bar{s}$ existe uma vez que $X \subset T$ é finito e evidentemente $\bar{s} \leq s$.

$\bar{s}$ é viável não supérflua. De fato, se $\bar{s}$ é supérflua, então por definição existe s* viável tal que $s^* \leq \bar{s}$, $s^* \neq \bar{s}$. Como $\bar{s} \leq s$, então $s^* \leq s$ e portanto $s^* \in X$.

Mas $p(s^*) < p(\bar{s})$, uma vez que p_i é estritamente crescente, o que contradiz a definição de $\bar{s}$. ||

O lema (10) garante a existência de pelo menos uma solução para o problema (Pk), desde que exista alguma configura - ção viável.

O lema seguinte é utilizado na demonstração da pro priedade de acessibilidade.(12)

11 Lema: Se s é uma configuração viável não supérflua e $s' \leq s (s' \neq s)$ então existe $s'' \in \Gamma(s')$ tal que $s'' \leq s$.

Demonstração : Seja s uma configuração viável não supérflua. Seja $s' \leq s$ $(s' \neq s)$.

Pela definição de configurações viáveis não supér- fluas, tem-se que s' é não viável. Logo, pelo corolário (7) , todo corte mínimo em s' tem capacidade menor que D .

Seja $K = H(s')$ o corte mínimo de s' que gera as sucessoras de s' (Ver definição de Γ em (8)). Então :

$$r_i \sum_{\varepsilon K} c_i^0 + s_i' \Delta c_i = c(K,s')$$

$$< D \quad \text{pois} \quad s' \quad \text{é não viável}$$

$$\leq c(K,s) \quad \text{pois} \quad s \quad \text{é viável}$$

$$= \sum_{r_i \varepsilon K} c_i^0 + s_i \Delta c_i$$

Logo, $\sum_{r_i \varepsilon K} (s_i - s_i') \Delta c_i > 0$ e como $\Delta c_i \geq 0$ tem- se:

$$\exists\, r_j \in K \quad \text{tal que} \quad s_j > s_j'$$

Seja $s'' = s' + e^j$, então

$s'' \leq s$ pois $s' \leq s$ e $s'_j < s_j$.

Portanto, $s'' \in \Gamma(s')$ pois $r_j \in K$. ||

Diz-se que uma configuração s é <u>acessível a partir de s^0</u> se existe em G um caminho ligando s^0 a s .

12 <u>Propriedade de acessibilidade</u> : Se s é uma configuração viável não supérflua, então s é acessível a partir de s^0 .

<u>Demonstração</u> : Seja s uma configuração viável não supérflua , $s^0 \leq s$.

Se $s^0 = s$, então a propriedade é válida.

Se $s^0 \neq s$ então pelo lema (11), existe $s' \in \Gamma(s^0)$ tal que $s' \leq s$.

Seja $(s^0, s^1, \ldots, s^j)$ um caminho em G tal que

13 a) $s^{i+1} \in \Gamma(s^i)$, $i=0,1,\ldots,j$

14 b) $s^i \leq s$, $i=0,1,\ldots,j$

onde s^j é a configuração de maior custo entre todas as configurações que satisfazem (13) e (14). s^j existe pois T é finito.

De (14) tem-se $s^j \leq s$.

Se $s^j \neq s$, então pelo lema (11) existe $s^{j+1} \in \Gamma(s^j)$ tal que $s^{j+1} \leq s$. Logo s^{j+1} satisfaz (13) e (14) e $p(s^{j+1}) > p(s^j)$ o que é absurdo.

Portanto, $s^j = s$ e $(s^0, s^1, \ldots, s^j)$ é um caminho em G de s^0 a s . ||

A propriedade de acessibilidade garante que existem caminhos em G ligando s^0 a todas as configurações viáveis não supérfluas. Como o custo de todos os caminhos em G ligando s^0 a uma mesma configuração é igual ao custo desta configuração, os k caminhos de s^0 às k configurações solução de (Pk) são k caminhos de menores custos de s^0 a configurações distintas de T*.

Por outro lado, se existem caminhos de s^0 a todas as configurações viáveis não supérfluas, e o custo de um caminho em G ligando s^0 a uma configuração é igual ao custo desta configuração, então as k configurações terminais dos k caminhos obtidos através da resolução de (Pbk) constituem uma solução de (Pk).

Conclui-se portanto, que os problemas (Pk) e (Pbk) são equivalentes no sentido de que se $(s^0, s^1, \ldots, s^\ell)$ é um dos k caminhos de uma solução de (Pbk) então s^ℓ é uma das k configurações de menor custo de T*; e se s é uma das k configurações de uma solução de (Pk) então existe um caminho $(s^0, s^1, \ldots, s^\ell)$ de uma solução de (Pbk) tal que $s = s^\ell$.

15 Teste de viabilidade e obtenção das sucessoras de uma configuração

A modelagem apresentada para a reformulação do problema de busca é bastante eficiente; não só por gerar um grafo com um número reduzido de caminhos, como também por fornecer um processo simples para a verificação da viabilidade de uma configuração. Este processo é introduzido a seguir.

Através da proposição (6) sabe-se que uma configuração é viável se o valor do fluxo máximo nesta configuração é maior ou igual a D. Pode-se, fazendo uso deste conhecimento, testar a viabilidade das configurações.

Se for aplicado o algoritmo de rotulação (A12) para o cálculo do fluxo máximo obtém-se, além do teste de viabilidade, elementos para a determinação das sucessoras das configurações. De fato, como foi observado em Ford (62, pag. 18) o corte $(X, \bar{X})$, considerando-se X como o conjunto de nós rotulados e $\bar{X}$ como o conjunto de nós não rotulados no momento em que o algoritmo para, é um corte mínimo. Logo, pode-se utilizar o corte $(X,\bar{X})$ para a obtenção das sucessoras das configurações.

O método exposto permite portanto, além de testar a viabilidade das configurações, caracterizar uma função H que associa a cada configuração um corte mínimo. A eficiência do método pode ser aumentada se o fluxo máximo de cada configuração for armazenado, e o cálculo do fluxo máximo das sucessoras destas configurações partir deste fluxo.

Secção 2 - ALGORITMO Â

O algoritmo Â estudado nesta secção é uma adaptação do algoritmo de Dijkstra para a resolução do problema (Pk). A importância deste algoritmo se deve à sua razoável eficiência e por nele ser baseado o algoritmo de programação heurística apresentado no Capítulo IV.

Inicialmente apresenta-se o algoritmo de programação dinâmica por ser comumente utilizado em problemas de busca e discute-se a sua ineficiência em relação ao algoritmo Â.

Os três algoritmos que serão apresentados possuem basicamente a mesma estrutura, diferindo apenas em alguns detalhes que serão observados no devido momento. Todos eles geram um subgrafo parcial do grafo G, por uma conveniente aplicação sucessiva do operador sucessor definido em (8).

Estrutura básica dos algoritmos

A expansão de uma configuração s consiste em encontrar o conjunto $\Gamma(s)$ de sucessores de s definido em (8).

Os algoritmos que serão apresentados manipulam duas listas de nós: lista aberto, cujos nós ainda não foram expandidos e lista fechado, cujos nós já foram expandidos.

Uma terceira lista, a lista final, é utilizada pelo algoritmo para armazenar as configurações viáveis já obtidas. Es

ta lista é limitada por k posições e deve conter, ao término do processo, uma solução do problema.

Apresenta-se a seguir o modelo básico dos algoritmos, levando-se em consideração as particularidades do grafo que serão comentadas posteriormente. O algoritmo utiliza uma função z que associa a cada nó da lista aberto um valor real. A escolha desta função diferencia os diversos algoritmos que seguem este modelo.

16 Algoritmo

Inicialmente as listas aberto e fechado estão vazias.

Passo 1 : Ponha s^0 em aberto, associando-lhe $z(s^0)$

Passo 2 : Verifique a regra de parada.

Passo 3 : Retire uma configuração s de aberto cujo valor z(s) é mínimo. Se s é viável, guarde s em final fazendo as eliminações segundo a regra de eliminações.
Vá para o passo 2.
Senão,vá para o passo 4.

Passo 4 : Ponha s em fechado e expanda s .
Para cada $s' \in \Gamma(s)$ se s' não está em aberto ou fechado, guarde-a em aberto associando-lhe z(s').
Vá para o passo 2.

17 Como no grafo G o custo de uma configuração independe do caminho, as comparações com as listas aberto e fechado po

dem ser simplificadas, dispensando-se comparações de custo. Pela mesma razão nos algoritmos de Dijkstra e programação dinâmica são dispensáveis as comparações com a lista fechado no passo 4. No algoritmo de programação heurística, estas comparações aumentam a eficiência exceto com heurísticas consistentes, o que será justificado em (IV.11).

Além das observações acima, os três algoritmos diferem no cálculo da função z e nas regras de parada e eliminações em final.

Aqui não há necessidade de apontadores, uma vez que não se está interessado no caminho percorrido para chegar a uma configuração do alvo e sim na própria configuração.

18 Algoritmo de programação dinâmica

O algoritmo de programação dinâmica para a resolução do problema (Pbk) possui a mesma estrutura do algoritmo (16) com as regras abaixo:

19 O valor z(s) associado a cada configuração s na lista aberto é dado por :

$$z(s) = \sum_{i=1}^{m} s_i$$

20 Regra de parada

O algoritmo para se a lista aberto está vazia.

21 Regra de eliminações

22 a) a configuração s' da lista final elimina a configura-

ção s se

$$s' \leq s$$

Se a lista final está completa então:

23 b) a configuração s elimina a configuração s' de final se

$$p(s') > p(s)$$

e s' é a configuração de maior custo em final.

24 c) a configuração s' de final elimina a configuração s se

$$p(s') \leq p(s)$$

e s' é a configuração de maior custo em final.

O valor z(s) definido em (19) estabelece em que estágio se encontra a configuração s ; ou seja, o número de ampliações feitas a partir de s^0 até a configuração s . O algoritmo retira da lista aberto a configuração na qual foi feito o menor número de acréscimos de capacidade.

Segundo a regra de parada, o algoritmo de programação dinâmica só deve parar quando a lista aberto está vazia. Isto é, depois de expandir todas as configurações não viáveis acessíveis a partir de s^0 . Note-se que devido à limitação do número de acréscimos de capacidade permitidos em cada ramo, o algoritmo para após um número finito de iterações.

O item (22) da regra de eliminações provoca a saída de configurações viáveis supérfluas da lista final. Os itens (23) e (24) eliminam as configurações de custo alto, fazendo com que ao término do algoritmo as k configurações da lista final sejam aquelas de menores custos.

A ineficiência deste algoritmo está na necessidade da expansão de todas as configurações não viáveis acessíveis para se chegar à solução do problema. Entretanto, este algoritmo é bom quando se está interessado em encontrar todas as configurações do alvo.

Algoritmo Â

Como o custo nos ramos de G são todos positivos, o algoritmo de Dijkstra pode ser utilizado para a busca do caminho de custo mínimo em G sem maiores dificuldades. (Ver (B6) e (B7)).

O algoritmo Â destina-se à determinação dos k caminhos de s^0 ao alvo (ou todos, caso não existam k) de menores custos. Para isto o algoritmo Â funciona como o algoritmo de Dijkstra sendo que após encontrar um caminho de custo mínimo, continua determinando caminhos de s^0 ao alvo até encontrar k caminhos (ou todos, caso não existam k).

Como foi observado em (17) não há necessidade de eliminações por custo na lista aberto. As eliminações em aberto serão feitas apenas por igualdade a fim de evitar a expansão de u

ma configuração mais de uma vez.

O algoritmo Â possui a mesma estrutura do algoritmo (16) com as seguintes regras:

25 A função z é definida por

$$z(s) = p(s)$$

onde p(s) é o custo associado à configuração s .

26 Regra de parada

O algoritmo para quando a lista final está completa, ou seja, suas k posições estão preenchidas; ou a lista aberto está vazia.

27 Regra de eliminações

A configuração s' em final elimina a configuração s se

$$s' \leq s.$$

Como o valor atribuido z(s) a cada configuração s é igual ao custo de s , o algoritmo Â , como o algoritmo de Dijkstra, expande apenas as configurações de custos baixos, deixando de lado momentaneamente aquelas de custos muito altos que não parecem ser promissoras para a resolução do problema. Portanto, o número de configurações expandidas pelo algoritmo Â pode ser bem menor que o número de expansões feitas pelo algoritmo de

programação dinâmica.

A regra de eliminações não permite a permanência de configurações supérfluas na lista final (Ver teorema IV.4), dispensando a necessidade de se verificar se uma configuração pertence ou não ao alvo que, como já foi comentado, seria um processo bastante dispendioso.

Como as configurações expandidas pelo algoritmo são aquelas de menores custos, quando a lista final está completa pode-se garantir que as configurações ali armazenadas são as viáveis não supérfluas de menores custos, o que não acontece com o algoritmo de programação dinâmica.

Se não existem k configurações no alvo, então a lista aberto fica vazia antes da lista final estar completa. Daí a necessidade de parar o algoritmo se a lista aberto está vazia. Nesse caso o número de expansões do algoritmo Â é igual ao de Programação Dinâmica.

A seguir apresenta-se o algoritmo Â com as regras já embutidas, com excessão da regra de eliminações.

30 Algoritmo Â

Inicialmente as listas aberto, fechado e final estão vazias.

Passo 1 : Ponha s^0 em aberto , associando-lhe $p(s^0) = 0$

Passo 2 : Se final está completa ou aberto vazia. Pare
Senão vá para o passo 3.

Passo 3 : Retire de aberto a configuração s de menor custo em aberto e passe-a para fechado .
Se s é viável, guarde-a em final fazendo as eliminações segundo a regra de eliminações.
Vá para o passo 2.
Senão vá para o passo 4.

Passo 4 : Expanda s . Para cada $s' \in \Gamma(s)$ se s' não está em aberto, guarde-a em aberto associando-lhe $p(s')$.
Vá para o passo 2.

O algoritmo Â é uma particularização do algoritmo de programação heurística A apresentado no Capítulo IV e a demonstração de que Â encontra uma solução do problema (Pbk) é uma consequência do teorema (IV.4) e do lema (IV.8).

A seguir apresenta-se um exemplo onde resolveu- se o problema (Pk) através dos dois algoritmos introduzidos nesta secção.

Exemplo I : Na rede R da figura 1 formulou-se o problema (Pk) para k=2 e custos lineares.

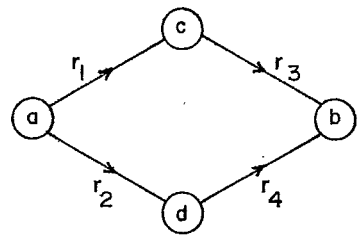


figura 1

Dados do problema :

$c^0 = (2,2,3,5)$ $\quad t = (4,4,4,4)$ $\quad D = 9$

$\Delta c = (1,2,3,2)$ $\quad p = (2,3,3,1)$ $\quad k = 2$

A solução encontrada para este problema é dada pe - las configurações da figura 2, com custos de ampliação $p(s^1) = 9$ e $p(s^2) = 10$ respectivamente.

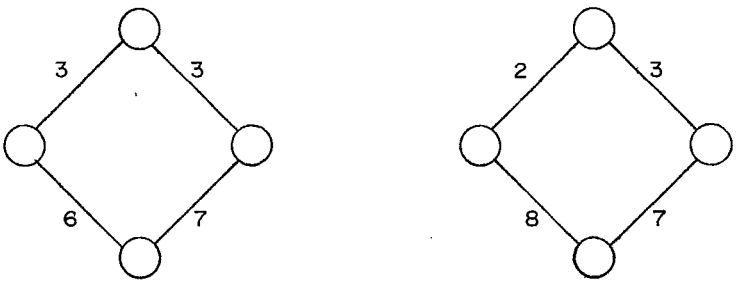


figura 2

A rede admite ainda as seguintes configurações viáveis não supérfluas:

$s^3 = (3,1,1,0)$ $\quad c(s^3) = (5,4,6,5)$ $\quad p(s^3) = 12$

$s^4 = (2,2,1,0)$ $\quad c(s^4) = (4,6,6,5)$ $\quad p(s^4) = 13$

A figura 3 apresenta o subgrafo parcial de G gerado pelo algoritmo de programação dinâmica. A figura 4 apresenta a parte do grafo gerada pelo algoritmo Â .

Programação dinâmica

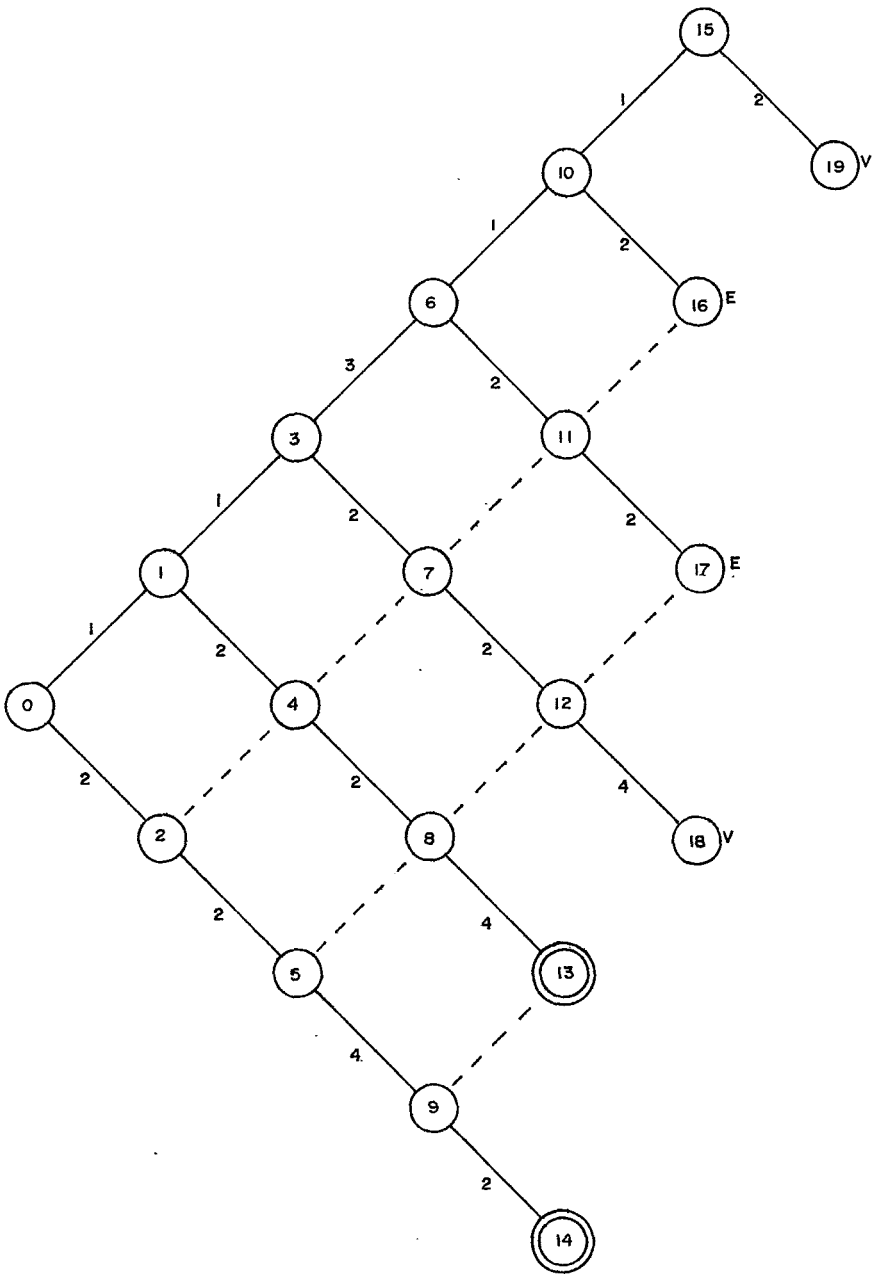


figura 3

(K) nó fechado no estágio K

((K)) viável não supérflua de menor custo

$(K)^E$ viável não supérflua que foi eliminada

(K)^{V} viável supérflua

i acréscimo no ramo r_i

Algoritmo Â

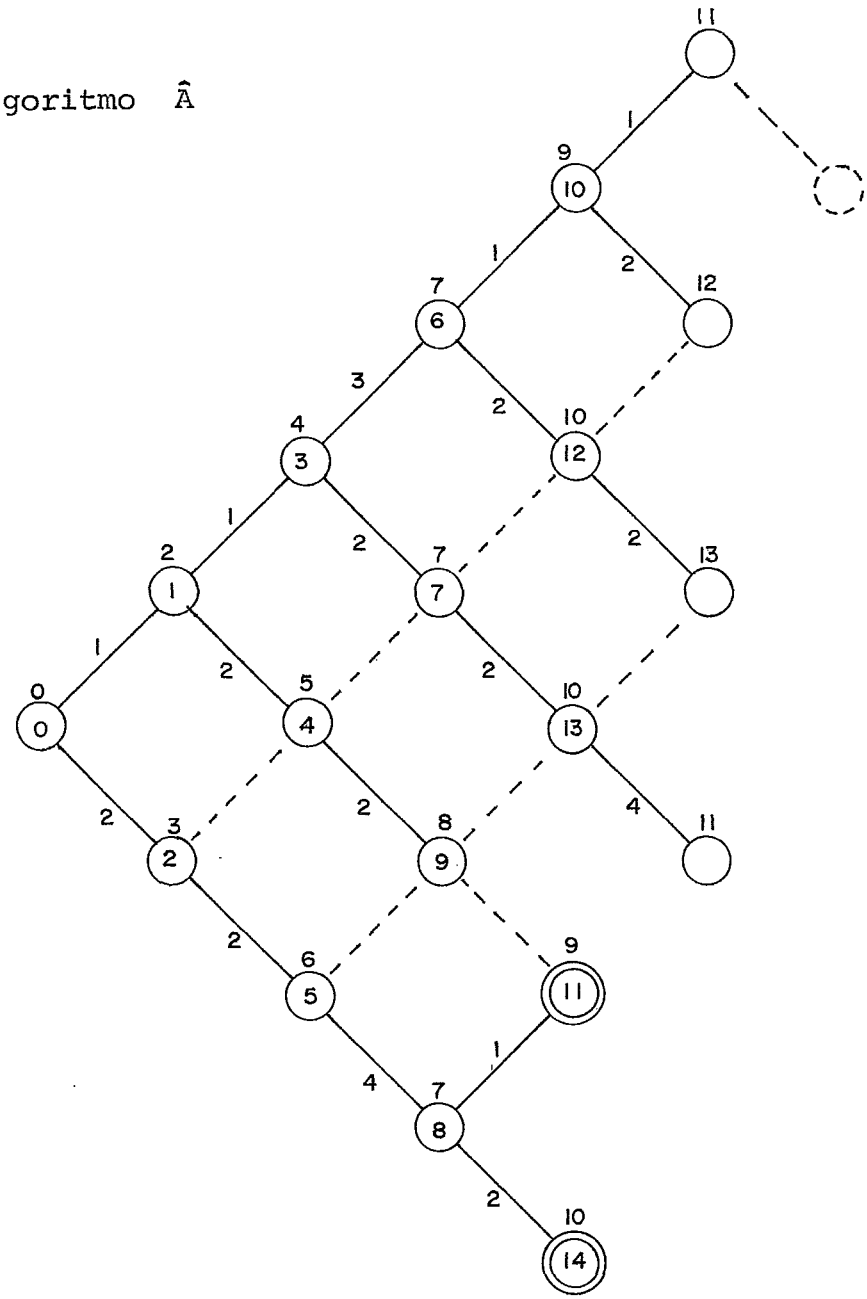


figura 4

(K) nó não gerado

(K) nó em aberto

(K)$^{\ell}$ nó em fechado no estágio K com custo ℓ

(K) viável não supérflua em final

i acréscimo no ramo r_i

Conclusões

Diante da dificuldade da resolução do problema (Pk) através de técnicas de programação inteira, reformulou-se este problema como o problema de busca (Pbk).

Como já foi comentado em (15) a modelagem apresentada para a reformulação do problema fornece resultados bastante interessantes: através da análise da estrutura da rede foi possível definir-se um operador sucessor que gera um grafo não muito extenso, o que reduz bastante a busca.

No exemplo I observa-se que devido ao limite de acréscimos t existem 625 configurações possíveis para o problema, entre as quais 269 são não viáveis, 356 são viáveis e 4 são viáveis não supérfluas. O operador sucessor entretanto, gera um grafo com 48 nós dos quais 14 são configurações não viáveis. O algoritmo de programação dinâmica expande 20 nós entre os 48 obtidos pelo operador sucessor. O algoritmo Â com k=1 expande 11 nós e com k=2 expande 14 nós na pior das hipóteses (coincidência de custos mínimos na lista aberto). Ver figuras 3 e 4 .

No algoritmo Â destinado à resolução de (Pbk), pode-se observar que a lista fechado é perfeitamente dispensável. De fato, no algoritmo de Dijkstra esta lista é utilizada para armazenar os nós do grafo já expandidos, o que posteriormente permite a recuperação do caminho ótimo. No problema (Pbk), porém, os cami - nhos percorridos para se chegar às configurações do alvo podem ser encontrados observando-se os acréscimos presentes em cada configu-

ração; logo, a lista fechado poderia ser dispensada o que em ter - mos computacionais significaria uma economia de memória.

Por outro lado, pode-se aumentar a eficiência da o- peração de Â utilizando-se o método proposto em (15) e destinan- do a lista fechado para armazenagem do fluxo máximo das configura- ções expandidas em lugar das configurações. Neste caso guardar-se- ia na lista aberto as sucessoras e apontadores que permitiriam bus car na lista fechado o fluxo máximo das configurações anteriores.

CAPÍTULO IV

O ALGORITMO DE PROGRAMAÇÃO HEURÍSTICA

Introdução

O algoritmo A* (Ver Hart |68|e (B8)) para a resolução do problema de encontrar um caminho de custo mínimo em um grafo, dependendo da heurística utilizada pode ser mais econômico em termos de trabalho computacional que o algoritmo de Dijkstra.

O algoritmo A , destinado à resolução do problema (Pk), apresentado neste capítulo é uma particularização do algoritmo A* . O algoritmo A, como o algoritmo A*, utiliza-se de informações heurísticas a respeito do problema. Este algoritmo após encontrar o caminho de custo mínimo de s^0 ao alvo, continua determinando caminhos de baixos custos até que k caminhos (ou todos, caso não existam k) de menores custos ligando s^0 ao alvo no grafo G , sejam obtidos.

Na secção 1 apresenta-se o algoritmo A e demonstra-se que o problema (Pk) pode ser resolvido através deste algoritmo.

Na secção 2 define-se uma heurística admissível que será utilizada pelo algoritmo e apresenta-se um método para o cál

culo desta heurística. No final desta secção resolve-se o problema do exemplo (III.1) através do algoritmo A e observa-se a eficiência deste algoritmo em relação aos algoritmos apresentados no capítulo III.

Secção 1 - O Algoritmo A

O algoritmo A que será estudado utiliza-se da teoria de programação heurística cuja idéia é dada a seguir.

O algoritmo Â do capítulo III expande as configurações geradas de menores custos. Suponha-se que para cada configuração s da rede R seja conhecido o custo mínimo $h(s)$ para a obtenção, a partir de s, de uma configuração do alvo. Supõe-se $h(s) = +\infty$ se não existe um caminho de s ao alvo. Então o custo do caminho ótimo de s^0 ao alvo é dado por

1 $$\min \{p(s) + h(s) \mid s \in T\}$$

Se a cada configuração s da rede R estiver associado o custo $p(s)+h(s)$ então o algoritmo A expande apenas as configurações pertencentes a caminhos ótimos, e a solução do problema (P1) é encontrada diretamente. Ou seja, expande-se o menor número de configurações necessárias para atingir-se o alvo.

Como no caso do problema (Pk) não existem meios pa

ra a determinação dos custos $h(s)$ antes da resolução do problema, considera-se uma estimativa $\hat{h}(s)$ do custo $h(s)$ denominada heurística de s .

A heurística de cada configuração s será determinada através de informações relacionadas com o custo dos caminhos ligando s ao alvo no grafo G definido por (III.1) e (III.8).

A heurística $\hat{h}$ é definida como uma aplicação de T em $\mathbb{R}$ que a cada configuração $s \in T$ associa uma estimativa do custo do caminho ótimo de s ao alvo.

O algoritmo $\hat{A}$ apresentado no capítulo III é o próprio algoritmo A no caso particular em que

$$\hat{h}(s) = 0 \quad , \quad \forall s \in T$$

Em Hart |68| demonstra-se que se $\hat{h}$ é uma heurística admissível (B9) e G é um δ-grafo, então o algoritmo A* é admissível (B10). A demonstração da admissibilidade de A* para grafos finitos sem circuitos de custos negativos encontra-se em Gonzaga . |71|.

A heurística $\hat{h}$ utilizada pelo algoritmo A é uma heurística admissível. O funcionamento do algoritmo A só está garantido se for considerada uma heurística deste tipo. A informação heurística a respeito do custo do caminho ótimo de uma configuração $s \in T$ ao alvo é utilizada pelo algoritmo através da função $z : T \rightarrow \mathbb{R}$ dada por

$$z(s) = p(s) + \hat{h}(s)$$

onde $p(s)$ é o custo da configuração s e $\hat{h}$ é uma heurística admissível.

$z(s)$ é portanto a estimativa do custo do caminho ótimo de s^0 a T^* contendo s.

O algoritmo A possui a mesma estrutura do algoritmo (III-16) e suas regras são idênticas às regras do algoritmo $\hat{A}$ (Ver(III.26) e(III.27)), com excessão da função z que aqui é dada por $p(s) + \hat{h}(s)$.

A seguir apresenta-se o algoritmo A com as regras já embutidas com excessão da regra de eliminações(III.27).

Algoritmo A :

Inicialmente as listas aberto, fechado e final estão vazias.

Passo 1 : Ponha s^0 em aberto associando-lhe $z(s^0) = p(s^0)+\hat{h}(s^0)$

Passo 2 : Se final está completa ou aberto vazia, pare.
Senão, vá para o passo 3.

Passo 3 : Retire de aberto a configuração s cujo valor $z(s)$ é mínimo em aberto. Passe-a para fechado.
Se s é viável, guarde s em final fazendo as eliminações segundo a regra de eliminações.
Vá para o passo 2.
Senão vá para o passo 4.

Passo 4 : Expanda s . Para cada $s' \in \Gamma(s)$ se s' não está em aberto ou fechado, guarde-a em aberto associando-lhe .

$z(s') = p(s') + \hat{h}(s')$.

Vá para o passo 2.

No algoritmo A* (B8) um nó após ser fechado pode ser reaberto, bastando para isso encontrar-se posteriormente um outro caminho até este nó de custo menor que o custo do caminho encontrado no momento em que ele foi fechado. O algoritmo A entretanto, não necessita reabrir um nó fechado devido à identidade de custos entre caminhos que ligam duas configurações.

A comparação no passo 4, entre a configuração s' que acabou de ser gerada e a lista fechado, também é feita no algoritmo A* e foi conservada no algoritmo A por aumentar a eficiência do algoritmo. Esta comparação impede a expansão da mesma configuração mais de uma vez no algoritmo A .

O algoritmo A resolve (Pbk)

O algoritmo A para k=1 é uma particularização do algoritmo A*, sendo que A devido às particularidades do grafo G não necessita reabrir um nó fechado.

Como o grafo G é um grafo finito sem circuitos de custos negativos (os custos dos ramos de G são todos positivos), a admissibilidade de A para k=1 é decorrente da admissibilidade de A* para grafos finitos sem circuitos de custos negativos.

O lema apresentado a seguir é uma adaptação do lema 1 da referência Hart |68| para o algoritmo A quando aplicado ao grafo G que possui características bastante particulares.

O lema 1 é bastante importante pois dele decorre uma série de resultados da teoria de programação heurística. O lema que será apresentado aqui é utilizado na demonstração da proposição 3 e mais adiante na demonstração do lema 8.

2 Lema : Seja s uma configuração acessível em G, não fechada no estágio ℓ do algoritmo A . Seja $P = (s^0, s^1, \ldots, s)$ um caminho em G ligando s^0 a s . Então existe $\bar{s} \in P$ tal que $\bar{s}$ está na lista aberto no estágio ℓ do algoritmo A .

A demonstração do lema 1 apresentada em Hart |68| considera o algoritmo A* e um δ-grafo. Devido ao fato de considerar-se um δ-grafo o caminho P ligando o nó inicial n^0 ao nó não fechado n , presente no enunciado do lema 2, deve ser um caminho ótimo ligando n^0 a n . Demonstra-se no lema 1 que com as hipóteses acima existe um nó $\bar{n} \in P$ em aberto, tal que, o caminho ligando n^0 a $\bar{n}$ encontrado pelo algoritmo A* é um caminho ótimo de n^0 a $\bar{n}$.

No caso do grafo G entretanto a demonstração de que o caminho ligando s^0 a $\bar{s}$ encontrado pelo algoritmo A é

um caminho ótimo de s^0 a $\bar{s}$ é dispensável devido à identidade de custos entre caminhos que ligam duas configurações em G . Pela mesma razão, considera-se o caminho P ligando s^0 a s como um caminho qualquer de s^0 a s .

A proposição seguinte será utilizada na demonstração de que o algoritmo A resolve o problema (Pk) quando $k > 1$.

3 Proposição : Suponha-se que o algoritmo A utiliza uma heurística admissível. Sejam s^1 configuração viável não supérflua e s^2 configuração viável, tais que $p(s^1) < p(s^2)$. Então s^2 não entra em fechado antes de s^1 .

Demonstração : Seja $\hat{h}$ uma heurística admissível utilizada pelo algoritmo A .

Suponha-se por absurdo que s^2 entra em fechado antes de s^1 .

Como s^1 é viável não supérflua, pela propriedade (III.12), s^1 é acessível em G . Seja $P = (s^{01}, s^{11}, \ldots\ldots s^{r1} = s^1)$ um caminho ligando s^0 a s^1 em G .

Por hipótese, s^2 entra na lista fechado antes de s^1 , portanto, na iteração em que s^2 entra em fechado s^1 não está em fechado. Logo pelo lema 2 existe uma configuração $\bar{s} \in P$ em aberto na iteração em que s^2 é fechada.

Como $\hat{h}$ é admissível,

$$z(\bar{s}) = p(s) + \hat{h}(\bar{s})$$
$$\leq p(s^1) \qquad \text{pois } s^1 \in T^*$$
$$< p(s^1) \qquad \text{por hipótese}$$
$$\leq p(s^2) + \hat{h}(s^2) \qquad \text{pois } s^2 \text{ é viável}$$
$$= z(s^2)$$

Logo vale $z(\bar{s}) < z(s^2)$ e portanto pelo passo 3 do algoritmo A , s^2 não poderia ser fechada antes de $\bar{s}$.

||

4 Teorema: O algoritmo A resolve o problema (Pk) .

Demonstração : A demonstração deste teorema está dividida em três partes: o algoritmo pára após um número finito de iterações, as configurações armazenadas em final pertecem ao alvo, as configurações do alvo encontradas pelo algoritmo são as de menores custos.

1ª parte : O algoritmo pára após um número finito de iterações.

Seja T o conjunto de todas as configurações de R, então,

$$T = \{s \in \mathbb{N}^m \ / \ s \leq t\}$$

O conjunto T é finito pois $T \in \mathbb{N}^m$ e t_i , $i=1,\ldots,m$, é um número finito.

Logo, o número de nós de G é finito e pelo passo 4 do algoritmo A cada nó de G é listado uma só vez, ou seja, cada configuração é expandida apenas uma vez. Portanto o algoritmo pára após um número finito de iterações.

2ª parte : As configurações armazenadas em final pertencem ao alvo.

Seja s uma configuração viável supérflua. Suponha-se que s entra na lista final. Pelo lema (III.10) existe s' viável não supérflua tal que

$$s' \leq s \quad (s' \neq s).$$

Logo $p(s') < p(s)$.

Pela proposição 3, s não entra em fechado antes de s' . Como por hipótese s entra em final e como $s' \in T^*$, então pelo passo 3 do algoritmo A , s' entra em final antes de s . s' não é eliminada de final pois não existe $s'' \leq s'$ $(s'' \neq s')$, s" viável.

Pela regra de eliminações (III.27) s ao ser guardado em final é eliminada por s' que já estava em final. Logo

s não consta na lista final.

3ª parte : As configurações do alvo encontradas pelo algoritmo são as de menores custos .

Seja s uma configuração viável não supérflua, tal que

$$p(s) < p(s')$$

onde s' é a configuração de maior custo na lista final no término do algoritmo.

Pela proposição (3) s entra em fechado antes de s' e s não pode ter sido eliminada por ser viável não supérflua. Logo s está em final. ||

O conceito de consistência introduzido a seguir caracteriza uma classe de heurísticas admissíveis que podem levar a uma maior eficiência do algoritmo A . O conceito geral de consistência encontra-se em (B11).

Heurísticas Consistentes

6 Definição: Uma heurística $\hat{h}$ é consistente no grafo G se quaisquer que sejam s^1, $s^2 \in T$, tais que existe um caminho em G ligando s^1 a s^2 então ,

$$\hat{p}(s^1,s^2) + \hat{h}(s^2) \geq \hat{h}(s^1).$$

onde $\hat{h}$ é uma heurística admissível e $\hat{p}(s^1,s^2)$ é dado por (III.3).

A utilização de uma heurística consistente pelo algoritmo A* garante que se um nó do grafo foi fechado, então o caminho ótimo ligando o nó inicial até este nó já foi encontrado pelo algoritmo. Portanto não há necessidade de reabrir um nó fechado, o que torna o algoritmo mais eficiente. (Ver demonstração em Hart |68|).

No caso do grafo G entretanto, uma configuração após ser fechada não necessita ser reaberta, devido à identidade dos custos entre caminhos ligando duas configurações. Consequentemente a utilização de uma heurística consistente não altera em nada a eficiência do algoritmo neste sentido.

A utilização de uma heurística consistente pelo algoritmo A é vantajosa se for introduzida uma pequena modificação na regra de retirada de configurações da lista aberto. Podendo-se, neste caso, garantir que uma configuração após ser fechada não será gerada uma segunda vez, sendo portanto dispensável a comparação com a lista fechado no passo 4 do algoritmo.

A regra de retirada na lista aberto sofre a seguinte modificação :

7 Retire de aberto a configuração s cujo valor z(s) é mí-

nimo em aberto.

Em caso de empates, escolha a configuração s de menor custo $p(s)$.

8 Lema : Suponha-se que o algoritmo A utiliza uma heurística consistente e que a regra de retirada da lista aberto é dada por (7). Então se uma configuração s foi fechada em algum estágio do algoritmo, s não será gerada pelo algoritmo em nenhum estágio posterior.

Demonstração :

Suponha-se que a configuração s foi fechada no estágio ℓ do algoritmo A , e que em algum estágio posterior foi gerada pelo caminho $(s^0,s^1,\ldots,s^j)$ em G , onde $s \in \Gamma(s^j)$.

Pelo lema 2 existe $\bar{s}$ uma configuração do caminho $(s^0,s^1,\ldots,s^j)$, tal que $\bar{s}$ está em aberto no estágio ℓ .

$\bar{s} \leq s$ $(\bar{s} \neq s)$,pois $\bar{s}$ pertence ao caminho $(s^0,s^1,\ldots,s^j)$ e $s \in \Gamma(s^j)$.

Como s foi fechada no estágio ℓ então

9 $$\hat{f}(s) \leq \hat{f}(\bar{s})$$

Pela hipótese de consistência,

$$\hat{p}(\bar{s},s) + \hat{h}(s) \geq \hat{h}(\bar{s})$$

ou seja,

$$p(s) + \hat{h}(s) \geq p(\bar{s}) + \hat{h}(\bar{s})$$

Portanto,

$$\hat{f}(s) \geq \hat{f}(\bar{s}) \qquad (10)$$

Por (9) e (10), conclui-se que:

$$\hat{f}(s) = \hat{f}(\bar{s})$$

e por (7)

$p(s) \leq p(\bar{s})$, o que é absurdo pois $\bar{s} \leq s$ $(\bar{s} \neq s)$ e p é estritamente crescente. ||

Conclui-se que se o algoritmo A funciona com as hipóteses do lema (8) então a comparação no passo 4 do algoritmo, entre a configuração s' que acabou de ser gerada e a lista fechado é desnecessária, pois neste caso a configuração s' não consta na lista fechado.

Em termos computacionais, esta modificação no algoritmo resulta em uma economia de trabalho: as comparações de custos introduzidas no passo 3 são mais econômicas que as comparações entre configurações. Além disso economiza-se também em memória , pois a lista fechado neste caso pode ser dispensada.

O algoritmo A com a modificação introduzida e com a heurística consistente $\hat{h}=0$ é o próprio algoritmo $\hat{A}$ apresentado no capítulo III. A demonstração da admissibilidade de $\hat{A}$ é portanto uma consequência do teorema (4) e do lema (8).

Secção 2 - Uma Heurística Admissível para (Pbk)

A heurística $\hat{h}$ que será definida é admissível por construção. Para efeito de simplificação do cálculo de $\hat{h}$ considera-se as funções custos p_i , i=1,...,m , lineares .

A heurística $\hat{h}$ será determinada através da resolução de um problema de programação linear. O processo para se chegar à formulação deste problema é apresentado a seguir.

Heurística de uma configuração não viável

Considere-se uma configuração s não viável. Pelo corolário (III.7) pode-se concluir que existe pelo menos um corte da rede R cuja capacidade em s é inferior a D .

Seja $\alpha = \{K_1, K_2, \ldots, K_\ell\}$ o conjunto de todos os cortes de R cujas capacidades em s são inferiores a D .

Uma configuração s' obtida da ampliação de s é viável se e somente se as capacidades de todos os cortes de α em s' são maiores ou iguais a D . Ou seja, se e somente se s' satisfaz:

a) $$c(K_j, s^0) + \sum_{r_i \in K_j} s'_i \, \Delta c_i \geq D \qquad j=1,2,\ldots,\ell$$

b) $$s \leq s' \leq t$$

devido a $$c(K_j, s') = c(K_j, s^0) + \sum_{r_i \in K_j} s'_i \Delta c_i$$

Portanto uma configuração viável de custo mínimo obtida da ampliação de s, pode ser encontrada através da resolução do seguinte problema de programação inteira.

$$\text{PI} \qquad \min \sum_{i=1}^{m} p_i(s'_i - s_i)$$

sujeito a

$$c(K_j, s^0) + \sum_{r_i \in K_j} s'_i \Delta c_i \geq D \qquad j=1,\ldots,\ell$$

$$s \leq s' \leq t$$

$$s'_i \text{ inteiro} \quad , \quad i=1,\ldots,m$$

O problema (PI) é um problema de programação inteira com $\ell+2m$ restrições, onde ℓ é a cardinalidade do conjunto α.

O valor ótimo de (PI) é o custo mínimo da viabilização de s. Pode-se verificar que toda solução de (PI) é uma configuração viável não supérflua. Logo o valor ótimo do problema (PI) é o custo mínimo para a obtenção a partir de s de uma configuração do alvo, ou seja $h(s)$.

A dificuldade da resolução do problema (PI) deve-se a dois fatores: ao possível grande número de restrições e ao excessivo trabalho prévio para a determinação do conjunto α.

Para definir-se uma heurística no problema (Pbk) contudo, é suficiente uma estimativa (subestimando) do valor $h(s)$. Para a estimativa $\hat{h}(s)$ que será definida, ignora-se algumas restri-

ções do problema (PI) considerando-se um subconjunto α' do conjunto de cortes α, e abandonando-se as restrições s_i' inteiro. Um subconjunto do conjunto α pode ser obtido pelo procedimento descrito em (17).

O problema (PI) reduzido desta forma é um problema de programação linear com $\ell'+2m$ restrições, onde ℓ' é a cardinalidade do conjunto α' ($\ell' \leq \ell$). Em geral, a dimensão do problema reduzido é menor que a dimensão do problema original devido ao abandono de algumas restrições.

Seja $\alpha' = \{K_1', K_2', \ldots, K_{\ell'}'\} \subset \alpha$, formula-se o problema reduzido como segue:

$$\text{PL} \qquad \min \sum_{i=1}^{m} p_i (s_i' - s_i)$$

sujeito a

$$c(K_j', s^0) + \sum_{r_i \in K_j'} s_i' \Delta c_i \geq D \qquad j=1,\ldots,\ell'$$

$$s \leq s' \leq t \qquad s' \in \mathbb{R}^m$$

12 Define-se a heurística de s como sendo o valor ótimo do problema (PL) de s.

Observa-se que o problema (PI) é mais restrito que o problema (PL). Logo, o valor ótimo de (PI) é maior ou igual ao valor ótimo de (PL) e portanto,

13 $$\hat{h}(s) \leq h(s)$$

Define-se a heurística de uma configuração viável s como sendo $\hat{h}(s) = 0$. Como $h(s) = \infty$ se não existe um caminho de s a T* (incluindo-se neste caso as configurações viáveis supérfluas) conclui-se que a expressão (13) vale para toda configuração s de T . Logo $\hat{h}$ é uma heurística admissível.

Se aos problemas (PL) de s , $s \in T$ forem incluídas as restrições " s_i inteiro, i=1,...,m ", a heurística fornecida por este problema ainda é admissível e mais eficiente que a heurística definida em (12), pois, em geral, subestima menos o valor h(s). A não inclusão destas restrições permitem contudo que o problema possa ser resolvido pelo método simplex.

Resolução do problema (PL)

O problema (PL) pode ainda estar vinculado por um número muito grande de restrições. Para simplificar a resolução deste problema procura-se decompô-lo de tal forma que cada problema resultante da decomposição esteja vinculado por poucas restrições.

A definição introduzida a seguir sugere uma partição do conjunto de cortes α' que possibilita uma decomposição de (PL).

14 Definição : Dois cortes $\bar{K}$ e $\bar{\bar{K}}$ de R são conexos se existem cortes $K_1, K_2, \ldots, K_j$ de R tais que

a) $K_1 = \bar{K}$, $K_j = \bar{\bar{K}}$

b) $K_i \cap K_{i+1} \neq \emptyset$, $i=1,\ldots,j$

Diz-se que dois cortes são desconexos se não são conexos.

Exemplo I : Considerem-se os cortes

$K_1 = \{r_1, r_2\}$

$K_2 = \{r_2, r_3\}$

$K_3 = \{r_3, r_4\}$

$K_4 = \{r_1, r_4\}$

$K_5 = \{r_5, r_6\}$

da rede da figura 1.

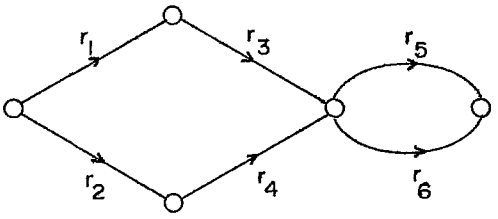


figura 1

Neste exemplo tem-se K_i é conexo de K_j para $i,j = 1,2,3,4$. O corte K_5 é conexo de si mesmo e desconexo de K_j , $j=1,2,3,4$.

A definição acima define uma partição $P(\alpha') = \{P_1, P_2,\ldots,P_q\}$ em α' caracterizada a seguir.

Tome-se um corte $K \in \alpha'$ e defina-se P_1 como sendo o conjunto de todos os cortes de α' conexos a K . Definidos os conjuntos $P_1,P_2,\ldots,P_r$, se $\bigcup_{i=1}^{r} P_i = \alpha'$ faz-se $q=r$; senão, toma-se $K \in \alpha'$ tal que $K \notin \bigcup_{i=1}^{r} P_i$ e define-se P_{r+1} como o conjunto de todos os cortes de α' conexos a K . A construção recorrente acima conclui-se, pois α' é finito.

Observa-se que com a definição dada , $P(\alpha')$ é uma partição de α' pois

$$P_i \neq \emptyset \qquad i = 1,\ldots,q$$

$$P_i \cap P_j = \emptyset \qquad i \neq j, \quad i,j=1,\ldots,q$$

$$\bigcup_{i=1}^{q} P_i = \alpha'$$

No exemplo I , $P(\alpha') = \{P_1,P_2\}$ é uma partição de α' onde

$$P_1 = \{K_1,K_2,K_3,K_4\}$$

$$P_2 = \{K_5\}$$

$P(\alpha')$ particiona os cortes de α' de tal forma

que cortes conexos pertençam a um mesmo P_j . Observa-se que os ramos pertencentes a cortes de P_i não pertencem a nenhum corte de P_j qualquer que seja $j=1,\ldots,q$, $j\neq i$.

A partição $P(\alpha')$ possibilita uma separação no conjunto de restrições de (PL) em q subconjuntos, cada um deles constituído de todas as restrições de (PL) provenientes de cortes de um P_j . Devido à definição dos P_j , $j=1,\ldots,q$, as variáveis s_i' presentes em um dos q subconjuntos de restrições não aparecem em nenhuma restrição não pertencente a este subconjunto. Além disso a função custo p é separável pois é linear. Logo, a minimização de PL pode ser feita por partes, o que pode ser observado no exemplo I. Neste exemplo pode-se minimizar a função custo considerando-se inicialmente os cortes de P_1 e posteriormente o corte K_5 de P_2 .

O problema (PL) será decomposto em q problemas do formato :

PLj $$\min \sum_{i=1}^{m} p_i(s_i'-s_i)$$

sujeito a

15 $$c(K,s^0) + \sum_{r_i \in K} s_i' \Delta c_i \geq D \qquad \forall K \in P_j$$

16 $$s_i \leq s_i' \leq t_i \quad , \quad s' \in \mathbb{R}^m$$

Considerando-se $X = \{r_i \in M \,/\, r_i \in K \;, \forall K \in P_j\}$, a dimensão do problema PLj é dada por

$$|X| + 2m \quad , \quad m \geq |X|$$

Observa-se entretanto que esta dimensão pode ser reduzida para $3|X|$, se for considerado o fato de $s'_i = s_i$ para todo $r_i \notin X$.

Pode-se verificar que encontrar uma solução do problema (PL) é equivalente a encontrar uma solução para cada (PLj) no sentido de que se

a) s^* é uma solução de (PL) então uma solução de (PLj) pode ser obtida por:

$$s_i^{*j} = \begin{cases} s_i^* & \text{se } r_i \in X \\ s_i & \text{se } r_i \notin X \end{cases}$$

b) $s^{*1}, s^{*2}, \ldots, s^{*q}$ são soluções de $PL_1, PL_2, \ldots, PL_q$ respectivamente então,

$s^* = s + \sum_{j=1}^{q} (s^{*j} - s)$ é uma solução de (PL).

Se a partição $P(\alpha')$ for unitária então a dimensão de (PL_1) é igual à dimensão de (PL) e neste caso a decomposição não contribui para a simplificação da resolução de (PL). Porém , no caso em que $P(\alpha')$ não é unitária, a dimensão de cada um dos (PLj) é menor que a dimensão de (PL) e portanto a resolução dos (PLj) é menos trabalhosa que a resolução do (PL).

Os problemas (PLj) podem ser resolvidos aplican -

do-se o método do simplex.

Para que a resolução dos (PLj) de s seja possível resta definir o subconjunto α' dos cortes de R , cujas capacidades em s são inferiores a D . A seguir apresenta-se um processo para a determinação de um subconjunto α' de s .

17 Determinação do subconjunto α' de uma configuração de T

Observa-se que devido a toda configuração $s \in T$ ser maior ou igual à configuração s^0 , um corte cuja capacidade é inferior a D em s tem garantidamente capacidade inferior a D em s^0 .

O método proposto para a determinação de um subconjunto de cortes cujas capacidades são inferiores a D em uma configuração s de T , parte de um subconjunto de cortes cujas capacidades em s^0 são inferiores a D .

Denomina-se <u>incremental</u> a um caminho no grafo G ligando s^0 a uma configuração viável, obtido através de uma otimização passo a passo. A incremental é portanto um caminho $(s^0, s^1, \ldots, s)$ em G tal que

a) s é viável

b) $p(s^{i+1}) \leq p(s')$, $\forall s' \in \Gamma(s^i)$

A determinação de uma incremental no grafo G for

nece um subconjunto de cortes α' da configuração s^0.

Define-se α' de s^0 como sendo o conjunto constituído pelos cortes utilizados para a expansão das configurações não viáveis da incremental.

O subconjunto α' de uma configuração $s \in T$ será obtido selecionando-se entre os cortes de α' de s^0 aqueles cujas capacidades em s são inferiores a D.

18 Obtenção da incremental

Inicialmente faça $s = s^0$. (*) Se s é viável, então o caminho encontrado ligando s^0 a s é a incremental; se não, expanda s. Escolha entre as sucessoras de s a configuração s^i de menor custo. Faça $s^i = s$ e volte para (*).

A figura 2 ilustra a incremental e o conjunto α' de s^0 do grafo do exemplo (III.1).

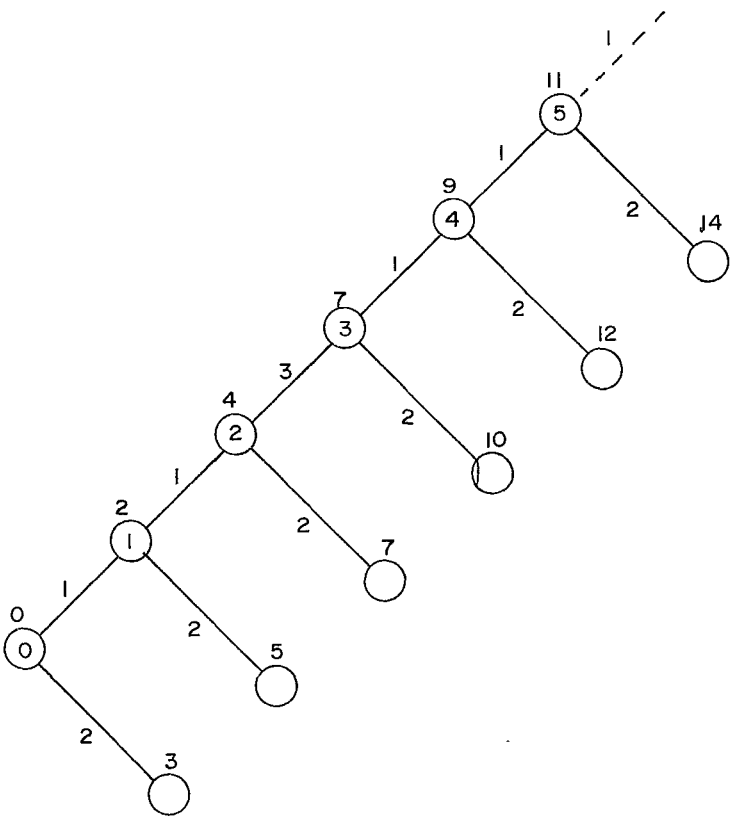


figura 2

O subconjunto α' de s^0 obtido é dado por

$$\alpha' = \{\{r_1, r_2\}, \{r_2, r_3\}\}$$

Observa-se que quando expandiu-se a quarta configuração houve um expate de custos: a configuração superior foi escolhida. Se a escolha recaisse sobre a outra configuração obter-se-ia o seguinte conjunto α' :

$$\alpha' = \{\{r_1, r_2\}, \{r_2, r_3\}, \{r_3, r_4\}\}$$

A heurística $\hat{h}$ definida por (12), quando determinada a partir dos conjuntos α' obtidos pelo procedimento descrito em (17), é uma heurística consistente.

Para a demonstração da consistência de $\hat{h}$ mostra-se inicialmente que se s^1 e s^2 são duas configurações de T tais que existe um caminho em G ligando s^1 a s^2, e se $\bar{s}$ é uma solução do PL de s^1 e $\bar{\bar{s}}$ é uma solução do PL de s^2 então,

$$p(\bar{\bar{s}}) \geq p(\bar{s}).$$

Considere-se X o conjunto de pontos viáveis do PL de s^1 e Y o conjunto de pontos viáveis do PL de s^2. Como existe um caminho em G de s^1 a s^2 então $s^1 \leq s^2$.

Afirmação : $Y \subset X$

Uma configuração s' pertence a X se :

a) $s' \geq s^1$

b) qualquer que seja K pertencente a α' de s^1 então,

$$c(K,s^0) + \sum_{r_i \in K} s'_i \Delta c_i \geq D$$

Seja $s'' \in Y$. Então (a) é satisfeita pois $s'' \geq s^2 \geq s^1$.

Se K pertence a α' de s^1 então por definição de α',

19 K pertence a α' de s^2

ou

20 $c(K,s^2) \geq D$

De (19), tem-se :

$$c(K,s^0) + \sum_{r_i \in K} s''_i \Delta c_i \geq D \quad \text{pois} \quad s'' \in Y$$

De (20) resulta (21) pois $s'' \geq s^2$.

Mostrou-se portanto que qualquer que seja $s'' \in Y$ então $s'' \in X$. Ou seja, $Y \subset X$, podendo-se concluir que :

22 $$\min_{s' \in Y} p(s') \geq \min_{s' \in X} p(s')$$

A demonstração da consistência de $\hat{h}$ decorre de (22) pois se s^1 e s^2 são duas configurações de T tais que existe um caminho em G ligando s^1 a s^2 então :

$$\hat{p}(s^1,s^2)+\hat{h}(s^2) = p(s^2)-p(s^1)+ \min_{s' \in Y} \{p(s'-s^2)\} \quad \text{por definição de } \hat{h}(s^2)$$

$$= \min_{s' \in Y} \{p(s')\} - p(s^1)$$

$$\geq \min_{s' \in X} \{p(s')\} - p(s^1) \quad \text{pois vale (22)}$$

$$= \min \{p(s'-s^1)\}$$

$$= \hat{h}(s^1)$$

As figuras 3 e 4 ilustram as regiões do grafo geradas pelo algoritmo A quando aplicado ao problema do exemplo

(III.1). As heurísticas utilizadas foram determinadas a partir dos conjuntos de cortes $\alpha' = \{\{r_1,r_2\},\{r_2,r_3\}\}$ na figura 3 e do conjunto de cortes $\alpha' = \{\{r_1,r_2\},\{r_2,r_3\},\{r_3,r_4\}\}$ na figura 4.

Neste exemplo pode-se observar uma redução do número de configurações expandidas pelo algoritmo A quando comparado ao número de configurações expandidas pelos algoritmos apresentados no capítulo III, ficando comprovada assim a maior eficiência deste algoritmo em relação aos outros dois.

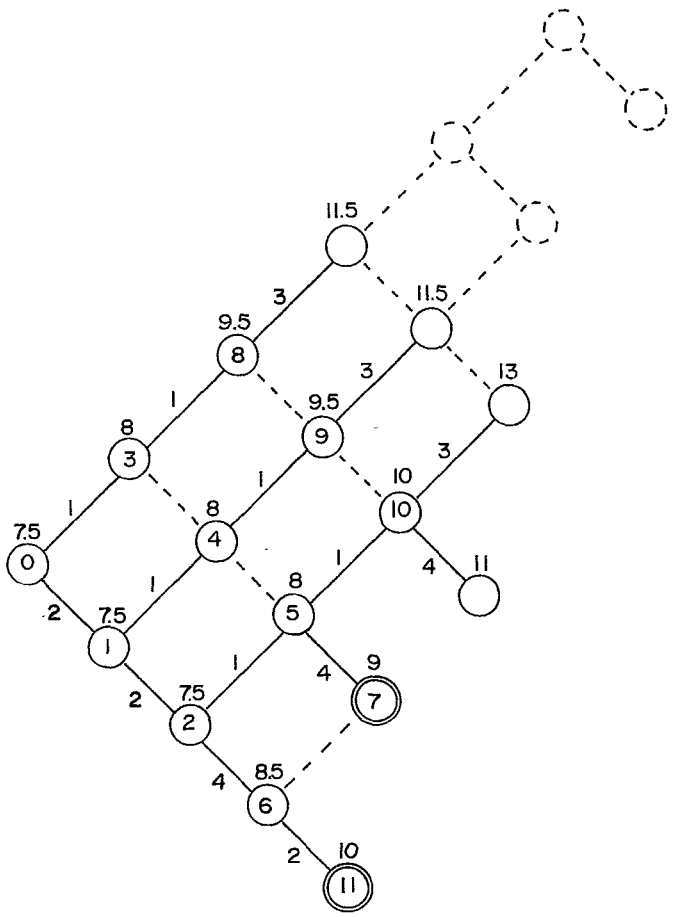


figura 3

nó não gerado

nó em aberto ℓ

nó em fechado no estágio K com custo heurístico ℓ

viável não supérflua em final ℓ

acréscimo no ramo r_i

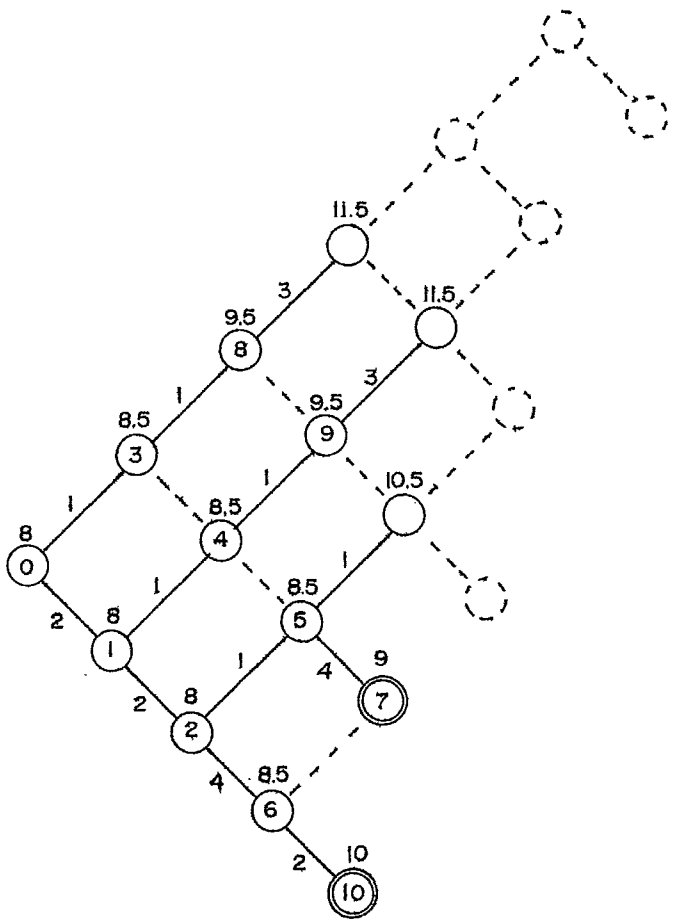


figura 4

Conclusões

A grande dificuldade para a aplicação do algoritmo A está na determinação de uma heurística admissível para o problema que não subestime excessivamente os valores $h(s)$, $s \in T$.

Uma heurística é tanto mais eficiente quanto mais próximo de $h(s)$ estiver o valor $\hat{h}(s)$, para todo s de T. A heurística definida em (12) depende diretamente do conjunto α' de cada configuração s. Por exemplo, se são conhecidos todos os cortes de R cujas capacidades em s são inferiores a D e se forem incluídas as restrições "s_i' inteiro, $i=1,2,\ldots,m$" ao problema (PL) de s então $\hat{h}(s) = h(s)$. Adotando-se esta heurística para todo s de T, as configurações expandidas pelo algoritmo A são aquelas pertencentes a caminhos ótimos e portanto expande-se o menor número possível de configurações. Conclui-se que o algoritmo A é tanto mais eficiente quanto maior for o número de cortes do conjunto α' de s^0.

Este fato pode ser observado nas figuras 3 e 4 quando resolveu-se o problema do exemplo (III.1) utilizando-se inicialmente a heurística determinada a partir dos cortes $\{r_1,r_2\},\{r_2,r_3\}$ e posteriormente a heurística determinada considerando-se os cortes $\{r_1,r_2\},\{r_2,r_3\},\{r_3,r_4\}$. Em ambos os casos o algoritmo expande 7 configurações para $k=1$. Para $k=2$, entretanto, o algoritmo expande 11 configurações no primeiro caso e 10 no segundo caso; reduzindo em 1 o número de configurações expandidas e em 2 o número de configurações geradas.

Como a heurística $\hat{h}$, definida em (12), além de ser admissível satisfaz a consistência, então pode-se melhorar a eficiência do algoritmo A introduzindo as modificações sugeridas quando da definição de heurísticas consistentes.

Na literatura encontram-se vários métodos para a resolução do problema de encontrar os k melhores caminhos em um grafo. Nas referências Polack |61| e Yen |71| apresentam-se alguns destes métodos. Justifica-se a não utilização destes algoritmos no presente trabalho por serem pouco eficientes quando se trata da resolução do problema (Pbk).

Bibliografia

Berge, C., Ghoila-Houri, A.

"Programmes, Jeux et Reseaux de Transport", Dunod, Paris, 1962.

Dijkstra, E.W.

"A Note on Two Problems in Connexion with Graphs", Numerische Mathematik , Vol. 1, pag. 269-271, 1959.

Ford, L. e Fulkerson

"Flows in Networks" , Princeton University Press,Princeton, 1962.

Gonzaga, C.C.

"Estudo de Algoritmos de Busca em Grafos e sua Aplicação a Problemas de Planejamento", Tese de Doutorado, COPPE, 1973.

Hart, P. Nilsson, N. e Raphael, B.

"A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Trans. Syst. So. Cybernetics, Vol. 4, Nº 2, pag. 100-107, Julho 1968.

Hu, T.C.

"Integer Programming and Network Flows", Addison Wesley,1969.

Pollack, M.

The k-th Best Route through a Network",Operations Research,

Vol. 9, Nº 4, pag. 578-580, 1961.

Yen, J.Y.

"Finding the k Shortest Loopless Paths in a Network", Management Science , Vol. 17, Nº 11, 1971.

APÊNDICE A

CONCEITOS DE TEORIA DOS GRAFOS

Neste Apêndice apresentam-se alguns conceitos básicos de teoria dos grafos utilizados neste trabalho. Nas referências Ford |62| e Berge |62| encontra-se um estudo mais detalhado desta teoria.

1 Uma rede orientada R é um par ordenado (N,M) onde N é o conjunto de nós de R e M é uma família, $M = (r_i)_{i \in I}$ cujos elementos os ramos de R satisfazem:

a) $r_i \in N \times N \quad , \; i \in I \subset \mathbb{N}$

b) $\forall \; n \in N \quad , \quad (n,n) \notin M$

A orientação de R fica estabelecida através da orientação dos ramos de R . Por exemplo, se $(n_1,n_2) \in M$ e $(n_2, n_1) \in M$ então (n_1,n_2) e (n_2,n_1) são considerados ramos distintos de R .

Um ramo $r_i \in M$ é um par ordenado (n_1,n_2) onde $n_1 \in N$ e $n_2 \in N$. n_1 é dito extremidade inicial do ramo r_i e n_2 extremidade final do ramo r_i .

Observa-se que é permitida a existência de ramos tais que $r_j = r_k \quad , \; j \neq k$; ou seja, ramos múltiplos, uma vez que

M é uma família. Em Ford |62| faz-se a restrição da não exis - tência de ramos múltiplos em R ,porém, a teoria lá desenvolvida é válida também para este caso.

Um caminho em R é uma sequência de ramos tais que a extremidade final de qualquer ramo intermediário coincide com a extremidade inicial do ramo seguinte.

O comprimento de um caminho é o número de ramos que o compõe.

Um circuito é um caminho onde a extremidade final do último ramo coincide com a extremidade inicial do primeiro.

Uma cadeia ou caminho não orientado é uma sequên - cia de ramos $r_1, r_2, \ldots, r_j$ tal que um ramo intermediário é ligado ao ramo anterior por uma das extremidades, e ao seguinte pela ou - tra. Se as extremidades de r_1 e r_j não ligadas a r_2 e r_{j-1} coin cidem, a cadeia recebe o nome de ciclo.

Neste trabalho considera-se uma rede orientada R na qual distinguem-se dois nós: a denominado fonte ou produtor e o nó b sumidouro ou consumidor.

À rede R estará associado o vetor capacidade $c \in \mathbb{N}^m$ onde c_i é a capacidade do ramo $r_i \in M$.

Uma rede R pode ser representada por uma figura geométrica onde os nós são representados por círculos e os ramos por seguimentos orientados entre dois nós.

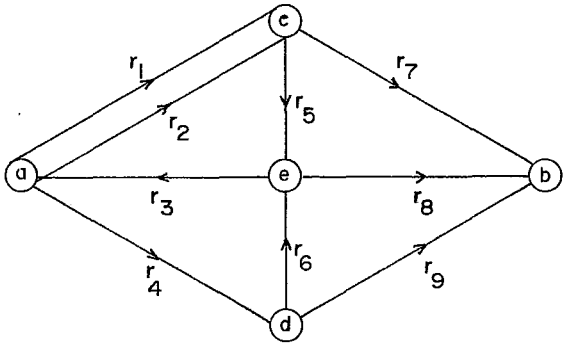
2 Exemplo : A figura seguinte representa a rede $R = (N,M)$ onde

$N = \{a,b,c,d,e\}$

$M = \{r_1, r_2, \ldots, r_8\}$

$r_1 = (a,c)$; $r_2 = (a,c)$; $r_3 = (e,a)$; $r_4 = (a,d)$; $r_5 = (c,e)$

$r_6 = (d,e)$; $r_7 = (c,b)$; $r_8 = (e,b)$; $r_9 = (d,b)$



Exemplos : caminho - (r_4, r_6, r_8)

circuito - (r_2, r_5, r_3)

cadeia - (r_3, r_5, r_7)

ciclo - (r_8, r_3, r_6)

3 <u>Notação</u>

Para simplificar a notação serão estabelecidas as seguintes convenções .

Sejam X e Y dois subconjuntos de N . Denote-se por (X,Y) o conjunto de todos os ramos (n_i, n_j) tais que $n_i \in X$ e $n_j \in Y$.

Se g é uma função de M em $\mathbb{R}$ define-se

4 $$g(X,Y) = \sum_{(n_i,n_j)\,\varepsilon\,(X,Y)} g(n_i,n_j) = \sum_{r_\ell\,\varepsilon\,(X,Y)} g(r_\ell)$$

Um <u>fluxo</u> de a para b em R é uma função de M em $\mathbb{N}$ tal que

5 $$f(n,N) - f(N,n) = 0 \qquad \forall\, n \,\varepsilon\, N\,,\quad n \neq a,b$$

O <u>valor do fluxo</u> f na rede R é dado por :

6 $$v(f) = f(a,N) - f(N,a)$$

A expressão (5) representa a lei de conservação do fluxo nos nós; ou seja, para todo nó $n \,\varepsilon\, N$ $n \neq a,b$, a soma dos fluxos dos ramos que entram em n é igual à soma dos fluxos que saem de n .

Pode-se verificar a partir de (5) e (6) que :

$$v(f) = f(N,b) - f(b,N)$$

Um fluxo <u>f em R é viável em relação a c</u> se

7 $$0 \leq f(r_i) \leq c_i \qquad \forall\, r_i \,\varepsilon\, M$$

onde c é o vetor capacidade associado a R .

8 Um <u>corte K de R</u> é um conjunto $(X,\bar{X})$ onde X e $\bar{X}$ satisfazem

$$a \,\varepsilon\, X \quad , \quad b \,\varepsilon\, \bar{X}$$

$$X \cap \bar{X} = \emptyset$$

$$X \cup \bar{X} = N$$

9 A capacidade de um corte $K = (X,\bar{X})$ na rede R com vetor capacidade c é dada por :

$$c(K) = \sum_{r_i \varepsilon (X,\bar{X})} c_i$$

Diz-se que K é um corte mínimo em R se sua capacidade é mínima entre as capacidades de todos os cortes de R .

10 Problema do fluxo máximo

Dada a rede R e o vetor capacidade c determinar um fluxo viável f de maior valor entre todos os fluxos viáveis de R .

A uma solução deste problema denomina-se fluxo máximo de R .

11 Teorema do fluxo máximo - corte mínimo

O valor do fluxo máximo de R é igual à capacidade de um corte mínimo de R .

A demonstração deste teorema e os resultados obtidos a partir dele encontram-se em Ford |62|.

Algoritmo de rotulação

O algoritmo de rotulação destina-se à determinação do fluxo máximo de R , e sua versão original é apresentada em Ford |62|.

O algoritmo parte de um fluxo viável f e faz aumentos de fluxo até que isto não seja mais possível.

Inicialmente procura-se uma cadeia em R do nó a ao nó b onde é possível um aumento de fluxo; modifica-se o fluxo nesta cadeia. O algoritmo para quando não existe nenhuma cadeia deste tipo onde é possível um aumento de fluxo.

O algoritmo classifica os nós de R em três tipos:

Nós não rotulados

Nós abertos- aqueles rotulados e não expandidos

Nós fechados - aqueles rotulados e expandidos

Os nós recebem rótulos do tipo (s, n, ε) , onde :

s é um sinal; pode ser + ou -

n é a denominação de um nó

ε é um número inteiro positivo ou infinito

12 Algoritmo

Inicialmente aberto e fechado estão vazias.

É dado um fluxo viável f .

Passo 0 : Ponha a em aberto com rótulo (+, 0, ∞)

Passo 1 : Se aberto está vazia, pare.

Senão vá para o passo 2 .

Passo 2 : Se $\underline{b}$ está em aberto, vá para o passo 3.

Senão, retire arbitrariamente um nó $\bar{n}$ de aberto. Ponha $\bar{n}$ em fechado.

Seja $(\pm, m, \bar{\varepsilon})$ o rótulo de $\bar{n}$.

Rotule todo nó n não rotulado tal que

$$a_i = (\bar{n},n) \in M$$

$$f(\bar{n},n) < c_i$$

com rótulo $(\pm, \bar{n}, \varepsilon)$ onde

$$\varepsilon = \text{Min}\{\bar{\varepsilon}, c_i - f(\bar{n},n)\}$$

e ponha n em aberto.

Rotule todo nó n não rotulado tal que

$$(n,\bar{n}) \in M$$

$$f(n,\bar{n}) > 0$$

com rótulo $(-, \bar{n}, \varepsilon)$ onde

$$\varepsilon = \text{Min}\{\bar{\varepsilon}, f(n,\bar{n})\}$$

e ponha n em aberto .

Vá para o passo 1.

Passo 3 : Seja $(+, \bar{n}, \varepsilon^*)$ o rótulo de b .

Faça $n=b$.

Vá para o passo 4.

Passo 4 : Seja $(s, \bar{n}, \varepsilon)$ o rótulo de n

Se $s=+$ faça

$$f(\bar{n},n) = f(\bar{n},n) + \varepsilon^*$$

Se $s=-$ faça

$f(n,\bar{n}) = f(n,\bar{n}) - \varepsilon^*$

Se $\bar{n} \neq a$ faça $n = \bar{n}$

Vá para o passo 4

Senão retire todos os nós de aberto e fechado. Apague os rótulos.

Vá para o passo 0.

O algoritmo de rotulação , além do fluxo máximo da rede R , permite determinar um corte mínimo de R . Consideran - do-se a última rotulação feita e tomando-se X como o conjunto de nós rotulados e o conjunto $\bar{X} = N/X$ de nós não rotulados, prova - se que $(X,\bar{X})$ é um corte mínimo de R .

APÊNDICE B

BUSCA DO CAMINHO DE CUSTO MÍNIMO EM UM GRAFO

Neste apêndice apresentam-se dois algoritmos para a busca do caminho de custo mínimo em um grafo: o Algoritmo de Dijkstra e o Algoritmo de Programação Heurística. Nestes algoritmos foram baseados os algoritmos Â e A apresentados nos Capítulos III e IV.

Inicia-se este apêndice introduzindo-se a definição de grafo e apresentando-se alguns conceitos relativos a grafos.

1 Definição : Um grafo G é um par ordenado (T,Γ) onde T é o conjunto de nós de G e Γ é o operador sucessor que a cada nó $n \in T$ faz associar um subconjunto $\Gamma(n)$ de nós de T .

Uma rede R sem ramos múltiplos pode ser representada por um grafo.

2 Um ramo de G é um par ordenado (n,n') onde $n \in T$ e $n' \in \Gamma(n)$. Os nós de $\Gamma(n)$ são denominados sucessores de n .

Denomina-se expansão do nó n a obtenção dos sucessores de n .

Um grafo G é finito se T é finito.

Um caminho em G é uma sequência de nós de T

$(n_1, n_2, \ldots, n_j)$ onde $n_{i+1} \in \Gamma(n_i)$, $i=1,2,\ldots,j-1$.

3 Problema de busca

Considere-se um grafo $G = (T,\Gamma)$ e uma função custo $\bar{p} : NxN \rightarrow R^+$ que associa a cada ramo de G o custo $\bar{p}(n,n')$.

Se os custos de todos os ramos de um grafo G são maiores ou iguais a um dado valor positivo δ diz-se que G é um δ-grafo.

O custo do caminho $A = (n_1,n_2,\ldots,n_j)$ de G é dado por

4 $$\bar{p}(A) = \sum_{i=1}^{j} \bar{p}(n_i,n_{i+1})$$

5 Pb Dado um nó inicial $n_0 \in T$ e um conjunto alvo $T^* \subset T$, encontrar um caminho de n_0 a T^* em G, cujo custo é mínimo entre todos os caminhos de n_0 a T^*.

Um algoritmo de busca é dito admissível se garantidamente encontra em tempo finito uma solução para o problema Pb.

Algoritmo de Dijkstra

No algoritmo de Dijkstra utiliza-se duas listas de nós:

lista aberto - onde são armazenados os nós gerados e não expandidos.

lista fechado - onde são armazenados os nós expandidos.

A cada nó n do grafo o algoritmo associa um custo c_n e um apontador.

6 Algoritmo :

Inicialmente as listas aberto e fechado estão vazias.

Passo 0 : Ponha n_0 em aberto associando-lhe custo $c_0 = 0$

Passo 1 : Se aberto está vazia, pare (o problema não tem solução).
Escolha em aberto o nó n com custo mínimo em aberto.
Resolva empates arbitrariamente, dando preferência aos nós em T^*.
Se $n \in T^*$ ponha n em fechado.
Vá para o passo 4.
Senão ponha n em fechado.
Vá para o passo 2.

Passo 2 : Expanda n . Para cada $n_i \in \Gamma(n)$ calcule

$$c_{n_i} = c_n + \bar{p}(n,n_i)$$

Se n_i está em aberto com custo maior que o custo encontrado agora elimine n_i de aberto.
Vá para o passo 3.

Passo 3 : Ponha todo $n_i \in \Gamma(n)$ que não esteja em aberto ou fechado em aberto, associando-lhe custo c_{n_i} e apontador n .
Vá para o passo 1.

Passo 4 : Recupere o caminho ótimo de n_0 a T* de trás para frente utilizando os apontadores.

7 Um estudo detalhado do algoritmo de Dijkstra encontra-se em Dijkstra |59|. Este algoritmo é admissível sempre que aplicado a grafos finitos sem cricuitos de custos negativos ou a δ-grafos.

Algoritmo A*

O algoritmo de programação heurística A* utiliza uma função $\hat{f}$ denominada função custo heurístico que a cada nó n de T , faz associar uma estimativa do custo do caminho ótimo ligando n^0 a T* que contém n .

Na referência Hart |68| estuda-se detalhadamente a operação do algoritmo A* , e as condições que garantem sua admissibilidade. Uma função custo heurístico é definida como sendo

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n) \quad , \quad \forall n \in T$$

onde $\hat{g}(n)$ é o custo do caminho de s^0 a n encontrado pelo algoritmo e $\hat{h}(n)$ é uma estimativa (subestimada) do custo do caminho ótimo de n a T* , denominada heurística de n .

O algoritmo A* manipula duas listas de nós: lista aberto de nós não expandidos e lista fechado de nós expandidos. O algoritmo associa a cada nó do grafo o valor $\hat{f}(n)$ e um apontador.

8 Algoritmo

Inicialmente as listas aberto e fechado estão vazias.

Passo 0 : Ponha n_0 em aberto associando-lhe $\hat{f}(n_0)$.

Passo 1 : Se aberto está vazia, pare (com insucesso).
Senão, escolha em aberto o nó n com $\hat{f}(n)$ mínimo em aberto.
Resolva empates arbitrariamente dando preferência aos nós em T^*.
Se $n \in T^*$ ponha n em fechado.
Vá para o passo 4.
Senão ponha n em fechado.
Vá para o passo 2.

Passo 2 : Expanda n. Para cada $n_i \in \Gamma(n)$ determine $\hat{f}(n_i)$.
Se n_i está em fechado com custo heurístico maior que o custo encontrado agora, retire n_i desta lista.
Vá para o passo 3.

Passo 3 : Ponha todo $n_i \in \Gamma(n)$ que não está em fechado em aberto com custo heurístico $\hat{f}(n_i)$ e apontador n.
Vá para o passo 1.

Passo 4 : Recupere o caminho ótimo de n_0 a T^* de trás para frente utilizando os apontadores.

Diz-se que uma heurística é admissível se

$$\hat{h}(n) \leq h(n) \quad , \forall n \in T$$

onde $h(n)$ é o custo do caminho ótimo de n a T^*.

10 Teorema : Se $\hat{h}$ é uma heurística admissível para todo n de T , então A* é admissível.

A demonstração deste teorema encontra-se em Hart |68| onde supõe-se que G é um δ-grafo. Esta demonstração foi refeita em Gonzaga |71| para o caso de grafos finitos sem circuitos de cus - tos negativos.

11 Diz-se que uma heurística admissível $\hat{h}$ é consisten te se quaisquer que sejam n_1 e n_2 , nós de T , tais que existe um caminho no grafo ligando n_1 a n_2 então

$$h(n_1,n_2) + \hat{h}(n_2) \geq \hat{h}(n_1) \quad ,$$

onde $h(n_1,n_2)$ é o custo do caminho ótimo no grafo, ligando n_1 a n_2 .

Demonstra-se em Hart |68| que se o algoritmo A* u- tiliza uma heurística consistente, então um nó fechado pelo algo - ritmo nunca será reaberto, o que dispensa a comparação com a lis- ta fechado no passo 2.