COMPARAÇÃO DE MÉTODOS DE RESOLUÇÃO DE FLUXO ÓTIMO EM REDES

Este exemplar corresponde a redação final da tese devidamente corrigida e defendida pelo Sr. **RAFAEL CARLOS VELEZ BENITO** e aprovada pela Comissão Julgadora.

Campinas, 04 de Setembro de 1991

Prof. Dr. _____
CLOVIS PERIN FILHO

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, **UNICAMP**, como requisito parcial para obtenção do Título de Mestre em Matemática Aplicada

AGRADECIMENTOS

Ao *Prof. Dr. Clóvis Perin Filho*, pela orientação e dedicação constantes durante todo o desenvolvimento deste trabalho.

A Minha esposa *Nelly Bernal Aguilar* e minha filha *Kimberlyn Ghandy Vélez Bernal*, pelo constante apoio e compreensão.

A meus pais *Rafael Vélez Chávez* e *Fortunata Benito de Vélez*, pelo invalorável apoio moral.

A Capes e Cnpq pelo apoio financeiro, e a Unicamp pelo apoio à pesquisa.

ÍNDICE

CAPITULO I

1.1. INTRODUÇÃO

Desde o desenvolvimento do método simplex por George B. Dantzig em 1947, que os modelos lineares de fluxo em rede têm sido estudados extensivamente. O problema de fluxo de custo mínimo é um programa linear especial cuja estrutura favorece a simplificação de métodos de resolução.

Matematicamente uma rede G é um par de conjuntos (N,A), onde N é um conjunto finito não vazio de n nós e A é um conjunto finito de m arcos; um arco é um par ordenado de nós distintos. Além disto, tem-se os seguintes vetores: um n-vetor de demandas $b=(b_i : i \in N)$, tal que para cada nó i, $b_i<0$ indica um nó produtor de fluxo e $b_i>0$ indica um nó consumidor; um m-vetor de custos $c=(c_j : j \in A)$ associado aos arcos; e os m-vetores de canalização de fluxo $l=(l_j : j \in A)$ e $d=(d_j : j \in A)$.

Assim, o problema de fluxo de custo mínimo consiste em determinar um vetor de fluxo $x=(x_j : j \in A)$ que é solução ótima de

$$
\begin{array}{lll}
 & \text{Minimizar} & c^t x \\
(1) & \text{sujeito a} & \hat{A}\, x = b \\
 & & l \leq x \leq d
\end{array}
$$

onde $\hat{A}$ é a matriz de incidência nóxarco da rede G.

São muito importantes as seguintes classes de problemas de programação em redes, casos particulares de (1), com as características indicadas:

- *Problema de Transbordo*: d=∞ (vetor onde cada componente é igual a infinito).

- *Problema de Transporte*: d=∞ e cada nó é ou um nó produtor ou um nó consumidor; além disto, cada arco é direcionado

de um nó produtor para um nó consumidor, formando uma rede bipartida.

- *Problema da Designação*: problema de transporte no qual $b_i = \pm 1 \ \forall \ i \in N$, formando uma rede com igual número de nós produtores e consumidores.

- *Problema de circulação*: $b=0$

- *Problema do Fluxo Máximo no arco* k: $b=0$, $d_k = \infty$, $c_k = -1$, $c_j = 0 \ \forall \ j \neq k$.

- *Problema de Caminho Mínimo entre os nós* p, q: $b_p = 1$, $b_q = -1$, $b_i = 0 \ \forall \ i \neq p,q$.

Para cada um dos problemas acima há métodos específicos que exploram as particulares estruturas e cujo estudo e desenvolvimento pode ser encontrado na literatura clássica de programação linear [12, 39, 40].

É possível mostrar que o problema do fluxo de custo mínimo (1) pode sempre ser transformado em um novo problema de fluxo de custo mínimo com $l=0$. Neste trabalho vamos considerar o problema formulado com $l=0$.

1.2. HISTÓRICO.

Os métodos de otimização são muito utilizados na tomada de decisões.

Estes métodos procuram a solução de diversos problemas nas mais variadas áreas, através da formulação de um modelo matemático, o qual consta de uma função objetivo sujeita a restrições. Assim por exemplo, no caso da programação linear, a função objetivo e as restrições são lineares. Como exemplo de áreas de aplicações, podemos citar: transporte de materiais, designação de tarefas, alocação de recursos, planejamento de operação de sistemas, caminho mínimo, fluxo máximo, etc. [31, 39, 40].

No presente trabalho, nós estamos interessados nos métodos de solução de Programação Linear especializados ao problema de fluxo por uma rede a custo mínimo (FCM), pois ele tem-se aplicado muito a problemas tais como: Sistemas de produção-distribuição, sistemas de logística, sistemas de tráfego urbano, sistemas de comunicação, sistemas de fluxos em redes de encanamento, sistemas de localização de recursos, sistemas de fluxos em redes elétricas, etc.

Em 1975 a Academia de Ciências " Royal Swedish " atribuiu o prêmio Nobel em ciências econômicas em igual cota ao professor Leonid Kantorovich da USSR e ao professor Tjalling C. Koopmans da USA por suas contribuições à teoria de distribuição ótima de recursos. Embora estes distintos professores tenham investigado uma grande variedade de problemas de otimização, é interessante notar que ambos estão associados com alguns dos primeiros periódicos descrevendo problemas de fluxos em redes como nós conhecemos atualmente. Em 1939, Kantorovich discutiu uma classe de modelos de otimização com exemplos específicos. A idéia por trás de cada exemplo era o planejamento de uma elevada produção de bens com

base na utilização ótima dos recursos existentes. Um destes exemplos envolve a distribuição do fluxo de carga entre as diferentes rotas da rede da estrada de ferro de tal maneira que satisfaça às restrições de capacidade nas rotas enquanto minimiza o gasto de combustível.

Embora na prática o algoritmo simplex seja eficiente, teoricamente pode-se construir classes de problemas de programação linear [32, 2, 21], mostrando que o comportamento desse algoritmo no pior caso não é polinomial (é exponencial). Assim, desde a primeira publicação do método simplex por Dantzig, tem havido muitas tentativas para achar melhores maneiras de resolver problemas de programação linear. Alguns melhoramentos do método simplex podem ser encontrados em [26, 20], por exemplo. Entretanto, o esforço computacional permanece não polinomial. Com o intuito de reduzir este esforço computacional, têm sido propostos vários métodos com outras abordagens. Em 1979 o matemático russo Leonid G. Khachiyan, mostrou que o método para otimização convexa desenvolvida pelo matemático russo N. Z. Shor e outros pode ser adaptado para produzir o Método dos Elipsóides [30, 39, 6] para resolução de problemas de programação linear com complexidade $O(n^{4.5}L)$, onde L é o tamanho dos dados do problema em "bits". Em 1984, N. Karmarkar [29] desenvolveu o Método Projetivo, baixando em muito a complexidade teórica em relação ao método dos elipsóides. Ele obteve um limite superior para o número de iterações de O(nL), e um limite para o número de operações de $O(n^{3.5}L)$. Ao mesmo tempo, anunciou resultados computacionais superiores aos obtidos pelo método simplex. O algoritmo original utilizava um formato especial para o problema, hipóteses muito restritivas e era muito difícil. Em 1985 várias pessoas introduziram melhoramentos ao método e desenvolveram variantes para o problema em formato padrão, incluíndo os seguintes pesquisadores: Anstreicher,

Gay, De Ghellinck e Vial, Gonzaga, Todd e Burrel, Ye. Ao mesmo tempo uma simplificação radical do algoritmo gerou o método afim escala, que se acredita não polinomial. Esse algoritmo já existia havia vinte anos, publicado por Dikin na USSR e era desconhecido. Foi redescoberto simultaneamente por Vial, Barnes, Vanderbei, Meketon e Freeman, Cavalier e Soyster, Adler e outros [3,47,1]. Resultados essencialmente novos foram obtidos a partir de 1986, com o estudo de métodos de trajetória central. Um ponto pertence à trajetória central se maximiza o produto das variáveis em um conjunto de pontos viáveis de mesmo custo. A trajetória central foi inicialmente estudada por Bayer e Lagarias [4] e por Megiddo [36]. Renegar [42] desenvolveu um algoritmo deste tipo , obtendo pela primeira vez um limite de complexidade de $O(\sqrt{n}L)$ iterações, mas com complexidade em número de operações aritméticas igual à de Karmarkar. Em 1987, Vaidya [45], seguindo a metodologia de Renegar, reduziu este limite para $O(n^3L)$. Por um caminho independente, Gonzaga [23] obteve o mesmo limite simultaneamente, utilizando um algoritmo do tipo penalidades. Novos algoritmos foram gerados a partir destes, incluindo os de Koyima, Mizuno e Yoshise [34], de Monteiro e Adler [37]. Fizeram-se extensões para programação quadrática convexa, obtendo-se os mesmos limites de complexidade. Em 1989 surgem novos algoritmos que procuram acompanhar a trajetória central por passos longos, ao invés dos passos curtos, que são essenciais para as demonstrações de complexidade da ordem de $\sqrt{n}L$ nos métodos citados anteriormente. Em 1990 publicou-se um artigo por Resende e Veiga [43] onde se resolve o problema de Designação usando o método dual afim e se mostra que ele somente consegue competir com o simplex para problemas de grande porte ($4x10^4$ nós e $2x10^6$ arcos) e ainda usando processamento em paralelo.

1.3. TERMINOLOGIA DE REDES.

Seja $G = (N,A)$, um grafo conexo, onde $N = \{1..n\}$ é o conjunto de nós e $A = \{1..m\}$ é o conjunto de arcos; um arco j é um par ordenado de nós distintos (t_j,h_j) onde t_j é denominado a cauda ('tail') e h_j a cabeça ('head') do arco. Seja $\hat{A} = (\hat{a}_{ij})$ a matriz de incidência nóxarco associada a G; isto é:

$$\hat{a}_{ij} = \begin{cases} +1 & \text{se } i = t_j \\ -1 & \text{se } i = h_j \\ 0 & \text{caso contrário} \end{cases}$$

Pode-se mostrar que a matriz de incidência $\hat{A}$ não é de posto completo pois $\sum_{i}^{m} \hat{a}_{ij} = 0$, $j \in A$. Entretanto, a eliminação de uma linha qualquer de $\hat{A}$ transforma-a numa matriz de posto completo uma vez que a rede associada é conexa. Neste texto vamos denotar por A a matriz obtida de $\hat{A}$ pela eliminação de sua última linha (n-ésima linha). O nó n, associado a esta linha que foi eliminada, é denominado nó raiz.

A seguir serão dadas algumas definições que serão usadas posteriormente.

Uma rede é própria se $n \geq 2$ e $m \geq 1$. Uma rede $G' = [N',A']$ é uma subrede de G se $N' \subset N$ e $A' \subset A$. Se $N'=N$, então nós dizemos que G' é uma subrede geradora de G.

Seja $P=(s_0,e_1,s_1,e_2,\ldots,s_{n-1},e_n,s_n)$ uma sequência finita com $e_j \in ((s_{i-1}, s_i),(s_i, s_{i-1}))$. P é um caminho se $s_0,s_1,\ldots,s_n$ são nós distintos e é um ciclo se $s_1,s_2,\ldots,s_n$ são nós distintos e $s_0=s_n$. Portanto, os arcos, tanto de um caminho quanto de um ciclo, são distintos. O comprimento de um caminho ou ciclo é definido como o número de arcos que possui.

Uma rede G é acíclica, se não contém ciclos. Uma rede é conexa, se para todo par de nós distintos, existe um caminho conectando-os.

Uma árvore é uma rede acíclica conexa. Pode-se mostrar que para qualquer par de nós distintos existe um único caminho em uma árvore conectando-os. Uma árvore que é uma subrede geradora de G é chamada uma árvore geradora de G com nó raiz n, possui n-1 arcos e é denotada por T.

O arco predecessor de um nó i≠n é o único arco incidente em i no caminho de n para i em T. Este arco é denotado por w_i. Defina $w_n=0$ para indicar que a raiz não possui predecessor.

Existe uma correspondência biunívoca entre o conjunto de árvores geradoras de G e o conjunto de bases formadas com as colunas de $\hat{A}$. Cada uma destas bases é uma matriz triangular o que facilita a resolução de sistemas lineares que a envolvam; é comum o uso do método de substituições sucessivas nestes casos.

Pode-se mostrar que AA^t é uma matriz simétrica positiva definida e diagonalmente dominante; os elementos da diagonal são iguais ao grau (interno+externo) de cada nó e os demais elementos são 0, -1, -2, ... indicando o número de arcos que ligam os nós correspondentes à linha e à coluna.

1.4 FORMULAÇÃO.

A cada arco $j \in A$ é associado um coeficiente custo c_j e uma capacidade máxima de fluxo d_j. A cada nó i da rede é associado uma demanda b_i onde $b_i < 0$ indica um nó produtor de fluxo e $b_i > 0$ indica um nó consumidor. Estamos supondo que $d > 0$. A formulação do Problema de fluxo de custo mínimo, aqui considerado, consiste em determinar um vetor de fluxo $x = (x_j)$, solução ótima do problema abaixo,

$$
\begin{array}{lll}
 & \text{Minimizar} & c^t x \\
(P) & \text{Sujeito a} & A\,x = b \\
 & & 0 \le x \le d
\end{array}
$$

Desta forma, deseja-se determinar um vetor x satisfazendo $0 \leq x_j \leq d_j$, $j \in A$ e $\sum_j a_{ij} x_j = b_i$ $i \in N$, e que minimize o custo total $\sum_j c_j x_j$.

O problema dual associado é:

$$\begin{array}{lll} & \text{Maximizar} & b^t y - d^t v \\ (D) & \text{sujeito a} & A^t y - v + w = c \\ & & v, w \geq 0 \end{array}$$

Ou seja, deseja-se determinar vetores y, v, w satisfazendo $v_j \geq 0$ $j \in A$, $w_j \geq 0$ $j \in A$, $y_{h_j} - y_{t_j} - v_j + w_j = c_j$ $j \in A$ e que maximize $\sum_i b_i y_i - \sum_j d_j v_j$.

As condições de folgas complementares são:

$$(x_j - d_j)\, v_j = 0 \qquad \forall\ j \in A \qquad (2)$$

$$x_j\, w_j = 0 \qquad \forall\ j \in A \qquad (3)$$

Dado um vetor de preços $y = (y_i)$ defina para cada arco $j \in A$ o coeficiente de custo relativo $\bar{c}_j = c_j + y_{t_j} - y_{h_j}$. Usando esta notação, pode-se eliminar os vetores v, w do problema dual (D), fazendo:

$$\bar{c}_j = c_j + y_{t_j} - y_{h_j} = w_j - v_j$$

Ou seja,

$$v_j = -\min(0, \bar{c}_j)$$

$$w_j = +\max(0, \bar{c}_j)$$

Assim, as condições de folgas complementares podem ser

reescritas.

Para cada arco $j \in A$:

$$\begin{aligned} \bar{c}_j < 0 \quad &\Rightarrow \quad x_j = 0 \\ \bar{c}_j = 0 \quad &\Leftarrow \quad 0 < x_j < d_j \\ \bar{c}_j > 0 \quad &\Rightarrow \quad x_j = d_j \end{aligned} \tag{4}$$

Um fluxo x é dito ser factível se $Ax=b$, $0 \leq x \leq d$. Se existe y tal que (x,y) satisfaz as condições de folgas complementares então x é um fluxo ótimo.

1.5. OBJETIVOS

Este trabalho tem por objetivo realizar um estudo dos métodos afim-escala, implementar em microcomputador diferentes versões destes métodos para programas lineares em redes e realizar testes computacionais comparando o desempenho destes métodos com os métodos primal e dual simplex, "out-of-kilter", tanto ao nível de tempo de cpu quanto ao nível do número de iterações, bem como com relação ao espaço de memória requerido.

CAPITULO II

UM MÉTODO PRIMAL DUAL

Neste capítulo é discutida uma especialização do Método "out-of-Kilter" para Problemas Lineares em redes capacitadas, onde será visto que a estrutura especial da matriz de coeficientes permite uma implementação muito eficiente.

2.1. MÉTODO "OUT OF KILTER" [KENNINGTON [31]]

Seja $[N,A]$ uma rede orientada. Considere o problema de fluxo de custo mínimo

$$
\begin{array}{lll}
 & \text{Minimizar} & c^t x \\
(P) & \text{sujeito a} & A\,x = b \\
 & & 0 \le x \le d
\end{array}
$$

cujo problema dual é:

$$
\begin{array}{lll}
 & \text{Maximizar} & b^t y - d^t v \\
(D) & \text{Sujeito a} & A^t y - v + w = c \\
 & & v, w \ge 0
\end{array}
$$

As condições de otimalidade podem ser escritas como:
Se (x,y) satisfaz

$$A\,x = b \tag{1}$$

$$
\left\{
\begin{array}{lll}
\bar{c}_j < 0 & \Rightarrow & x_j = 0 \\
0 < x_j < d_j & \Rightarrow & \bar{c}_j = 0 \\
\bar{c}_j > 0 & \Rightarrow & x_j = d_j
\end{array}
\right\} \; j \in A \tag{2}
$$

onde $\bar{c}_j = c_j + y_{t_j} - y_{h_j} = w_j - v_j$, então x é um fluxo ótimo

para (P).

O método " **Out-of-Kilter** " começa com qualquer vetor de preços **y**, e um vetor de fluxos **x** factível; isto é, satisfaz **Ax = b**, $0 \leq x \leq d$. Se (x,y) satisfaz (2), nós terminamos pois **x** é ótimo; caso contrário, ou o fluxo **x**, ou preço **y**, ou ambos são alterados com o objetivo de satisfazer (2), mantendo a restrição (1) sempre satisfeita.

Dado (x,y), considere um arco qualquer j. Se (2) é satisfeito por j , então dizemos que j está " **IN-KILTER** "; caso contrário dizemos que j está " **OUT-OF-KILTER** ".
As diversas condições de "*Kilter*" que podem ocorrer estão apresentadas na tabela 1:

TABELA 1: Condições de "*Kilter*".

	$\bar{c}_j < 0$	$\bar{c}_j = 0$	$\bar{c}_j > 0$
$x_j = d_j$	*out-of-kilter*	**In-kilter**	**In-kilter**
$0 < x_j < d_j$	*out-of-kilter*	**In-kilter**	*out-of-kilter*
$x_j = 0$	**In-kilter**	**In-kilter**	*out-of-kilter*

Para cada uma destas 9 condições é definido o *número de Kilter*, como mostrado na tabela 2:

TABELA 2: Número de "*Kilter*"

	$\bar{c}_j < 0$	$\bar{c}_j = 0$	$\bar{c}_j > 0$
$x_j = d_j$	x_j	**0**	**0**
$0 < x_j < d_j$	x_j	**0**	$d_j - x_j$
$x_j = 0$	**0**	**0**	$d_j - x_j$

Observe que os números de *Kilter* são todos não negativos e representam a quantidade de fluxo requerida para que um dado arco se torne " **In-Kilter** ". Se K_j denota o número de *Kilter* do arco j, então o número de *Kilter* de uma solução (x,y), é dado por:

$$\sum_{j \in A} K_j .$$

Portanto, uma solução cujo número de *Kilter* é zero é uma solução ótima do problema (P).

O método " **OUT-OF-KILTER** " consiste de duas fases. A fase Primal produz mudanças no fluxo enquanto que a fase Dual produz mudanças nos preços, de tal forma que o número de *Kilter* diminua. Ou seja, este método executa alternadamente estas duas fases, a primal e a dual, até que :

$$\sum_{j \in A} K_j = 0$$

2.2 Fase Primal.

Durante a fase Primal as variáveis duais (preços) permanecem fixas. O fluxo que muda está sempre associado com arcos em um ciclo.

Seja $Q = \langle s_0, e_1, s_1, e_2, \ldots s_{k-1}, e_k, s_k \rangle$ um ciclo em $[N,A]$ e seja q um m-vetor indicador do ciclo Q; isto é

$$q_j = \begin{cases} +1 & \text{se } e_j \in Q \text{ com direção de } s_0 \text{ para } s_k \\ -1 & \text{se } e_j \in Q \text{ com direção oposta} \\ 0 & \text{se } e_j \notin Q \end{cases}$$

Desta forma, nós temos

$$A\,q = 0.$$

Se x denota qualquer fluxo tal que $A\,x = b$, então, para qualquer escalar λ e qualquer q indicador de um ciclo, $\hat{x} = x + \lambda q$ satisfaz $A\,\hat{x} = b$. Para $\lambda > 0$ nós dizemos que o fluxo é incrementado em Q de λ. Para um arco j tendo $q_j = +1$, $q_j = -1$, $q_j = 0$, o fluxo atual aumenta, diminui, não se altera; respectivamente.

Denote por K_j o número de *Kilter* do arco j com fluxo x_j e por $\hat{K}_j$ o número de Kilter do arco j com fluxo $\hat{x}_j$. A fase primal serve para encontrar um ciclo Q com indicador q tal que, para $\hat{x} = x + q$, $\hat{K}_j \leq K_j \ \forall\ j$, e $\hat{K}_k < K_k$ para pelo menos um arco k. Em outras palavras, nosso objetivo é encontrar um ciclo cujo aumento de fluxo diminua o número de *kilter* total.

Em resumo, a fase Primal consiste em: *Dado um arco qualquer que está* **"Out-of-Kilter"**, *digamos* s, *encontrar um ciclo que inclua* s *e que permita um aumento de fluxo.* Suponha que o arco selecionado tenha $\bar{c}_s > 0$. Logo uma estratégia para determinar quando tal ciclo existe envolve o crescimento de

uma árvore especial, digamos T, enraizada em t_s. Esta árvore é construída de tal maneira que, se o nó $i \in T$, então um aumento de fluxo é permitido no único caminho em T desde o nó i à raiz. Se $h_s \in T$, o ciclo é o único caminho em T desde h_s à t_s através do arco s.

A seguir apresentamos o *Algoritmo* para a fase Primal. Este Algoritmo é iniciado com o arco s "**Out-of-Kilter**".

Algoritmo 2.1. **Fase Primal**

Passo 1. (*Iniciação*).

Seja $s \in A$ um arco "**Out of kilter**"

se $\bar{c}_s < 0$ então $\hat{N} \leftarrow \{ h_s \}$ e $\Delta_{h_s} \leftarrow x_s$

senão $\hat{N} \leftarrow \{ t_s \}$ e $\Delta_{t_s} \leftarrow d_s - x_s$.

Faça $\hat{A} \leftarrow \phi$

Passo 2. (*Determine candidatos para a árvore*).

Seja:

$$\psi_1 = \{ j \neq s:\ \bar{c}_j \geq 0,\ x_j < d_j,\ t_j \notin \hat{N},\ h_j \in \hat{N} \}$$

$$\psi_2 = \{ j \neq s:\ \bar{c}_j \leq 0,\ x_j > 0,\ t_j \in \hat{N},\ h_j \notin \hat{N} \}$$

Se $\psi_1 \cup \psi_2 = \phi$, então termine com a fase Primal [ou seja não existe ciclo].

Passo 3. (*Adicione um novo arco à árvore*).

Seleciono arco $k \in (\psi_1 \cup \psi_2)$.

Se $k \in \psi_1$, então faça $\Delta_{t_k} \leftarrow\!- \text{Min}\left\{ \Delta_{h_k} , d_k - x_k \right\}$

Se $k \in \psi_2$, então faça $\Delta_{h_k} \leftarrow\!- \text{Min}\left\{ \Delta_{t_k} , x_k \right\}$

Faça : $\hat{N} \leftarrow\!- \hat{N} \cup \left\{ t_k, h_k \right\}$ e $\hat{A} \leftarrow\!- \hat{A} \cup \left\{ k \right\}$

Se $\left\{ t_s, h_s \right\} \subset \hat{N}$, então vá ao passo 4; caso contrário volte ao Passo 2.

Passo 4. (*Atualização do fluxo*).

Se $\bar{c}_s < 0$, então aumente o fluxo no ciclo em Δ_{t_s} ; caso contrário aumente o fluxo no ciclo por Δ_{h_s}.

2.3. Fase Dual:

Se a fase Primal termina com a conclusão que não existe ciclo, então nós devemos reduzir o número de kilter para tal solução, quer dizer $\sum K_j$, fazendo com que o fluxo permaneça fixo e ajustando as variáveis duais.

Seja $T=[\hat{N},\hat{A}]$ a árvore obtida na fase primal. O decréscimo permitido nos preços para um arco j com $h_j \in \hat{N}$ e $t_j \in N-\hat{N}$, é dado na tabela 3, enquanto que o decréscimo permitido nos preços para um arco j com $t_j \in \hat{N}$ e $h_j \in N-\hat{N}$ é dado na tabela 4. As áreas em branco em ambas tabelas correspondem as condições que não podem ocorrer.

TABELA 3: Decréscimo permitido em y_{h_j}

	$\bar{c}_j < 0$	$\bar{c}_j = 0$	$\bar{c}_j > 0$
$x_j = d_j$	$-\bar{c}_j$	∞	∞
$0 < x_j < d_j$	$-\bar{c}_j$		
$x_j = 0$	$-\bar{c}_j$		

TABELA 4: Decréscimo permitido em y_{t_j}

	$\bar{c}_j < 0$	$\bar{c}_j = 0$	$\bar{c}_j > 0$
$x_j = d_j$			$\bar{c}_j$
$0 < x_j < d_j$			$\bar{c}_j$
$x_j = 0$	∞	∞	$\bar{c}_j$

A seguir apresentamos o *Algoritmo* para a fase Dual. Este Algoritmo é iniciado com a árvore $T = [\hat{N},\hat{A}]$, obtida na fase Primal.

Algoritmo 2.2. **Fase Dual**

Passo 1. (*Determine um arco incidente em T*).

Seja:

$$\psi_1 = \left\{ j:\ h_j \in \hat{N},\ t_j \notin \hat{N},\ e\ \bar{c}_j < 0 \right\}$$

$$\psi_2 = \left\{ j:\ h_j \notin \hat{N},\ t_j \in \hat{N},\ e\ \bar{c}_j > 0 \right\}$$

Passo 2. (*Determine o máximo incremento permissível*).

Faça:

$$\delta = \underset{j\in(\psi_1 \cup \psi_2)}{\text{Minimo}} \left\{ |\ \bar{c}_j| \right\}$$

Passo 3. (*Atualização das variáveis duais*).

Faça: $y_i \leftarrow\!\!- y_i - \delta,$

Finalmente, o *Algoritmo* **Out-of-Kilter** será dado a seguir:

2.4. ALGORITMO.

O método **Out-of-Kilter** consiste de duas fases (*Primal e Dual*) encadeadas de acordo com o algoritmo abaixo.

Algoritmo 2.3. **OUT-OF-KILTER**

Passo 1. (*Iniciação*).

Seja **x** um vetor qualquer de fluxo tal que **A x** = b, **0** ≤ **x** ≤ **d**.

Seja **y** um vetor qualquer de preços.

Passo 2. (*Encontre um arco* "**Out-of-Kilter**").

Seja s um arco "**Out-of-Kilter**" qualquer. Se todos os arcos estão "**In-Kilter**", então **pare, x** é uma solução ótima; caso contrário vá para o passo 3.

Passo 3. (*Fase Primal*).

Execute o *Algoritmo* 2.1. com o arco s "**Out-of-Kilter**". Se o *Algoritmo* 2.1. termina com a com a conclusão que não existe ciclo, vá ao passo 4; caso contrário volte ao passo 2.

Passo 4. (*Fase Dual*).

Execute o *Algoritmo* 2.2. com a árvore obtida no passo 3. Se s está "**Out-of-Kilter**", então volte ao passo 3; caso contrário volte ao passo 2.

CAPITULO III

MÉTODO SIMPLEX

Neste capítulo são discutidas especializações dos Métodos Primal e Dual Simplex para Problemas Lineares em redes capacitadas, onde será visto que a estrutura especial da matriz de coeficientes permite uma implementação muito eficiente.

3.1. INTRODUÇÃO.

Considere o Programa Linear:

$$\begin{array}{lll} & \text{Minimizar} & c^t x \\ (P) & \text{Sujeito a} & A\,x = b \\ & & 0 \le x \le d \end{array}$$

onde **A** é uma matriz de posto completo.

O problema dual é:

$$\begin{array}{lll} & \text{Maximizar} & b^t y - d^t v \\ (D) & \text{sujeito a} & A^t y - v + w = c \\ & & v, w \ge 0 \end{array}$$

O método simplex trabalha apenas com soluções básicas, cada uma definida em função de uma base **B** formada por colunas de **A**. Suponha, sem perda de generalidade, que as colunas de **A** estão ordenadas de tal forma que:

$$\mathbf{A} = [\, \mathbf{B} \mid \mathbf{L} \mid \mathbf{U} \,]$$

onde **B** é não singular.

As variáveis associadas às colunas de **B** são denominadas básicas. A matriz **L** está associada às variáveis não básicas x_j

com valor fixado em $x_j=0$ e a matriz **U** está associada às variáveis não básicas x_j com valor fixado em $x_j=d_j$. As variáveis básicas tem seus valores determinadas pelo sistema $Ax = b$. Desta forma, cada partição $\mathbf{B}|\mathbf{L}|\mathbf{U}$ das colunas de **A**, onde **B** é não singular, define uma solução básica.

Os vetores x, c, **d**, **v**, **w** podem ser particionados de forma similar.

$$x = \begin{bmatrix} x_B \\ x_L \\ x_U \end{bmatrix} \qquad d = \begin{bmatrix} d_B \\ d_L \\ d_U \end{bmatrix} \qquad c = \begin{bmatrix} c_B \\ c_L \\ c_U \end{bmatrix} \qquad v = \begin{bmatrix} v_B \\ v_L \\ v_U \end{bmatrix} \qquad w = \begin{bmatrix} w_B \\ w_L \\ w_U \end{bmatrix}$$

Assim (**P**) pode ser reescrito

$$\text{Minimizar} \qquad c_B^t x_B + c_L^t x_L + c_U^t x_U$$

$$\text{sujeito a} \qquad \mathbf{B}\, x_B + \mathbf{L}\, x_L + \mathbf{U}\, x_U = b$$

$$0 \le x_B \le d_B$$
$$0 \le x_L \le d_L$$
$$0 \le x_U \le d_U$$

e o dual associado é

$$\text{Maximizar} \qquad b^t y - d^t v$$

$$\text{sujeito a} \qquad \mathbf{B}^t y - v_B + w_B = c_B$$
$$\mathbf{L}^t y - v_L + w_L = c_L$$
$$\mathbf{U}^t y - v_U + w_U = c_U$$
$$v, w \ge 0$$

Vamos denotar por **B** o conjunto de índices das variáveis

básicas; isto é, x_{B_i} é a i-ésima variável básica. Além disto L, U denota o conjunto de variáveis não básicas associadas às matrizes L, U. Trata-se de um abuso de notação que é bastante utilizado.

Uma solução básica (x,y) é obtida da seguinte forma.

$$
\begin{aligned}
x_L &= 0 \\
x_U &= d_U \\
x_B &= B^{-1}b - B^{-1}Lx_L - B^{-1}Ux_U \\
y &= B^{-t}c_B = (B^{-1})^t c_B
\end{aligned}
$$

de tal forma que

$$Ax = Bx_B + Lx_L + Ux_U$$

$$B^t y = c_B$$

Seja $\bar{c} = w - v = c - A^t y$

Uma solução básica é dita primal factível, se

$$0 \leq x_B \leq d_B$$

Uma solução básica é dita dual factível, se

$$\bar{c}_j < 0 \quad \Rightarrow \quad x_j = d_j \qquad \forall\ j \notin B$$

$$\bar{c}_j > 0 \quad \Rightarrow \quad x_j = 0 \qquad \forall\ j \notin B$$

Uma solução básica é ótima quando é primal e dual factível simultaneamente.

Dada uma rede G=(N,A), seja T uma árvore geradora em G com raiz no nó n. Existe uma relação biunívoca entre as árvores geradoras T e as bases B. O arco predecessor de um nó $i \neq n$ é o único arco incidente em i no caminho P_i de n para i em T. Este arco é denotado por w_i. Defina $w_n=0$. Assim o vetor w define as variáveis básicas. Um arco s é dito para baixo em T se s é um arco do caminho de n para h_s em T, ou seja, $i=h(w_i)$; e se $s \in T$ é um arco do caminho de t_s para n em T, s é dito um arco para cima em T, ou seja, $i=t(w_i)$. Denote por P_i^+ o conjunto de arcos para baixo de P_i e por P_i^- os arcos para cima. De modo similar defina T^+, T^-.

Dado um caminho ou ciclo C com direção definida por um dos seus arcos, diz-se que $j \in C$ é um arco para frente se a sua direção concorda com a do ciclo ou caminho, caso contrário o arco é dito para trás. Denote por C^+ o conjunto de arcos para frente e por C^- o conjunto de arcos para trás.

Seja T uma árvore geradora. Dado um arco $s \notin T$, define-se C_s como sendo o ciclo formado em $T \cup (s)$. Observe que $s \in C_s$.

Dado um arco $r \in T$, define-se a partição de nós (N_r, N'_r) induzida pelo arco r, onde $N_r = \{ i \in N: r \notin P_i \}$. Ou seja N_r é o conjunto de nós que permanecem conectados à raiz após a remoção do arco r da árvore geradora T. Além disto, dado um conjunto de nós $N \subset N$, define-se o conjunto de arcos com cauda em N e cabeça em $N'=N-N$ por $A(N) = \{ j \in A: t_j \in N, h_j \in N \}$. Assim, $A(N_r) \cup A(N'_r) \cup (r)$ forma o corte induzido pelo arco básico $r \in T$, que consiste do conjunto minimal de arcos cuja remoção juntamente com r desconecta G.

Dado um conjunto de nós N, defina $b(N) = \sum [b_i: i \in N]$. Dado um conjunto de arcos A, defina $x(A) = \sum [x_j: j \in A]$; d(A), c(A) são definidos de modo similar.

Assim, dada uma partição de arcos (T,L,U) a solução básica associada é

$$x_j = 0 \qquad j \in L$$

$$x_j = d_j \qquad j \in U$$

$$x_j = b(N_j) + d(U \cap A(N_j)) - d(U \cap A(N')), \qquad j \in T^-$$

$$x_j = -b(N_j) - d(U \cap A(N_j)) + d(U \cap A(N')), \qquad j \in T^+$$

Observe que dados uma partição de nós (N,N'), um fluxo x tal que Ax = b e um preço y as equações

$$x(A(N')) - x(A(N)) = b(N)$$

$$y_{h_j} - y_{t_j} = c_j \qquad j \in T, \text{ sempre são satisfeitas.}$$

3.2. METODO PRIMAL SIMPLEX CANALIZADO PARA REDES [31].

O método primal simplex para resolução de um problema de fluxo ótimo, é dado a seguir, supondo-se conhecida uma partição básica primal factível (T,L,U) com solução básica factível associada (x,y), para iniciar a aplicação do método:

Passo **1.** (*Iniciação*).

Seja (T,L,U) uma partição básica primal factível associada à solução básica primal factível (x,y).

Passo **2.** (*Otimalidade*)

Defina $\bar{c}_j = c_j + y_{t_j} - y_{h_j} \qquad \forall j \in T$

Seja $S = \{ j \in L : \bar{c}_j < 0 \} \cup \{ j \in U : \bar{c}_j > 0 \}$

Se $S = \phi$, **pare**: solução (x,y) é ótima.

Passo **3.** (*Pivoteamento*)

Escolha $s \in S$. Seja C_s o ciclo em $T \cup \{s\}$ com a direção dada por s se $x_s = 0$, e com a direção contrária, se $x_s = d_s$.

Determine $\delta = \min \{ x_j : j \in C_s^- \} \cup \{ d_j - x_j : j \in C_s^+ \}$

Escolha $r \in \{ j \in C_s^-, x_j = \delta \} \cup \{ j \in C_s^+, d_j - x_j = \delta \}$

Atualize

$$x_j \longleftarrow \begin{cases} x_j + \delta & \text{se } j \in C_s^+ \\ x_j - \delta & \text{se } j \in C_s^- \\ x_j & \text{caso contrário} \end{cases}$$

Faça $T \longleftarrow (T \cup \{s\}) - \{r\}$; se $x_r = 0$ então $L \longleftarrow L \cup \{r\}$, senão $U \longleftarrow U \cup \{r\}$; se $\bar{c}_s < 0$ então $L \longleftarrow L - \{s\}$, senão $U \longleftarrow U - \{s\}$.

Se $s \in T^+$ então faça $\Delta = \bar{c}_s$ senão $\Delta = -\bar{c}_s$

Atualize

$$y_i \longleftarrow \begin{cases} y_i & \text{se } i \in N_s \\ y_i + \Delta & \text{se } i \in N_s \end{cases}$$

vá para 2.

O método primal simplex trabalha com um vetor de fluxo factível x desde o início. A cada iteração o método procura melhorar o valor da função objetivo passando de um fluxo factível para outro até alcançar um fluxo ótimo ou constatar que o problema não possui fluxo ótimo finito.

A cada iteração é escolhido para entrar na base um arco candidato do conjunto $S = \{ j \in L : \bar{c}_j < 0 \} \cup \{ j \in U : \bar{c}_j > 0 \}$

No capítulo 5 é discutida uma estrutura de dados para implementar uma árvore geradora e realizar as atualizações em x, y, T, **L**, **U** apresentadas acima.

3.3 MÉTODO DUAL SIMPLEX CANALIZADO PARA REDES.

Enquanto o método primal simplex mantém a factibilidade primal, satisfaz as condições de folgas complementares e a cada pivoteamento procura reduzir a infactibilidade dual, o método dual simplex faz justamente o oposto. Ele mantém a factibilidade dual, satisfaz as condições de folgas complementares e a cada pivoteamento procura reduzir a infactibilidade primal.

O método dual simplex é bastante empregado em análise de pós-otimalidade, quando são feitas pequenas modificações nos parâmetros do modelo. Por exemplo, depois de obtida uma solução ótima, deseja-se alterar os vetores b, d ou deseja-se introduzir uma nova restrição. Além disto, muitas vezes é mais fácil começar com uma solução básica dual factível e procurar sua factibilidade primal, do que obter uma solução básica factível inicial e depois otimizá-la, como se faz no método primal simplex.

No caso do método dual simplex canalizado para redes, ele requer uma partição dual factível (T, L, U). Uma vez que todos os arcos não básicos são mantidos ou em seu limitante inferior ou superior, a infactibilidade primal ocorre somente quando os arcos básicos violam os limites da canalização. Assim, denota-se por R o conjunto dos arcos com fluxo violando os limites para uma base T, isto é,

$$R = \{\, j \in T \;:\; x_j < 0 \;\; U \;\; x_j > d_j \,\}$$

Portanto, qualquer arco pertencente a R é um candidato a deixar a base. Também o custo relativo para cada arco $j \in A$ é definido como $\bar{c}_j = c_j + y_{t_j} - y_{h_j}$ e Δ é definido como sendo a mudança de fluxo em $r \in R$.

A seguir será apresentado o *Método Dual Simplex*

Canalizado para Redes.

Passo **1.** (*Iniciação*).

Seja $(T,\ \mathbf{L},\ \mathbf{U})$ uma partição básica dual factível associada à solução básica (x,y).

Passo **2.** (*Otimalidade*)

Seja $R = \{\ j \in T\ :\ x_j < 0 \ \vee\ x_j > d_j\ \}$

Se $R = \phi$, **pare** ; solução (x,y) é ótima.

Passo **3.** (*Pivoteamento*)

Escolha $r \in R$ para sair da base.

- se $x_r > d_r$, então seja $\Delta = x_r - d_r$ e defina

$$S = (A(N_r)) \cap \mathbf{L}) \cup (A(N'_r)) \cap \mathbf{U})$$

- se $x_r < 0$, então seja $\Delta = x_r$ e defina

$$S = (A(N_r)) \cap \mathbf{U}) \cup (A(N'_r)) \cap \mathbf{L})$$

Se $S = \phi$, **pare**: solução ótima ilimitada (e primal infactível). Caso contrário, escolha $s = \text{argmin}\{\ |\bar{c}_j|\ :\ j \in S\ \}$. Considere o ciclo C_s com direção de s se $\bar{c}_s < 0$ e com direção contrária de s se $\bar{c}_s > 0$. Atualize:

$$\bar{x}_j \longleftarrow \begin{cases} x_j + \Delta, & \text{se } j \in C_s^+ \\ x_j - \Delta, & \text{se } j \in C_s^- \\ x_j, & \text{se } j \notin C_s \end{cases}$$

Se o novo valor de x_r é zero então $\mathbf{L} \leftarrow\!\!- \mathbf{L} \cup \{r\}$; senão $\mathbf{U} \leftarrow\!\!- \mathbf{U} \cup \{r\}$; se x_s estava em seu limite inferior, então $\mathbf{L} \leftarrow\!\!- \mathbf{L} - \{s\}$, senão $\mathbf{U} \leftarrow\!\!- \mathbf{U} - \{s\}$. Se $r \neq s$ faça $T \leftarrow\!\!- T \cup \{s\} - \{r\}$. Se $s \in T^+$ então faça $\delta = \bar{c}_s$ senão $\delta = -\bar{c}_s$; atualizo

$$y \leftarrow\!\!- \begin{cases} y_i & \text{se } i \in N_s \\ y_i + \delta & \text{se } i \notin N_s \end{cases}$$

vá para o passo 2.

No capítulo 5 é discutida uma estrutura de dados para implementar uma árvore geradora e realizar as atualizações em x, y, T, **L**, **U** apresentadas acima.

CAPITULO IV+

MÉTODOS DE PONTOS INTERIORES

4.1. INTRODUÇÃO.

Uma Transformação Linear $T : R^n \longrightarrow R^m$, definida em R^n com valores em R^m é representada por uma matriz $A \in R^{mxn}$. A uma matriz de transformação $A \in R^{mxn}$ associam-se dois espaços importantes:

(i) O espaço imagem de A:

$$I(A) = \left\{ y \in R^m \ / \ y = Ax, \ x \in R^n \right\},$$

que é o conjunto de todas as combinações lineares das colunas de A.

(ii) O espaço nulo de A:

$$N(A) = \left\{ x \in R^n \ / \ Ax = 0 \right\},$$

que é o conjunto de pontos de R^n que são mapeados na origem.

O *posto*(A) de uma transformação linear A é a dimensão de seu espaço imagem, igual ao número de colunas linearmente independentes (e também igual ao número de linhas linearmente independentes).

Uma matriz A é de posto completo se *posto*(A) = min(m, n). Neste trabalho vamos supor que m<n e que A é uma matriz de posto completo.

O espaço nulo e o espaço imagem de uma transformação $A \in R^{mxn}$ estão respectivamente em R^n e R^m.

Uma relação geométrica muito interessante existe entre N(A) e o espaço imagem da transposta de A, $I(A^t)$: esses dois espaços são sub-espaços ortogonais de R^n e geram o espaço todo, como veremos a seguir:

+ Somente neste capítulo a matriz A tem m linhas (nós) e n colunas (arcos).

Teorema: Um vetor $v \in R^n$ é ortogonal a $I(A^t) \Leftrightarrow v \in N(A)$.

Demonstração:

$v \perp I(A^t) \Leftrightarrow v \perp A_i^t$, $i = 1,\ldots,m$, onde A_i representam as linhas de A. A relação acima é equivalente a:

$A_i v = 0$, $i = 1,\ldots,m \Leftrightarrow Av = 0$. Por tanto $v \in N(A)$.

Deste **Teorema** conclui-se que R^n é a soma direta dos subespaços $N(A)$ e $I(A^t)$, isto é, dado qualquer vetor $c \in R^n$, ele pode ser expresso como:

$$c = \hat{c} + \bar{c}, \qquad \text{onde } \hat{c} \in N(A),\ \bar{c} \in I(A^t).$$

O vetor $\hat{c}$ é a *projeção* de c sobre $N(A)$ e $\bar{c}$ é o *complemento ortogonal* de c em relação ao espaço nulo de A.

Teorema: A matriz AA^t é não singular.

Demonstração: (Por absurdo)

Suponha que para algum $y \neq 0$, $AA^t y = 0$.

Pré-multiplicando esta expressão por y^t, temos:

$y^t AA^t y = 0$ ou $(A^t y)^t (A^t y) = 0$

Esta última expressão é equivalente a:

$||A^t y||_2 = 0 \Leftrightarrow A^t y = 0$. O qual é impossível, posto que as colunas de A^t são linearmente independentes. Portanto AA^t é não singular. Completando a demonstração.

Podemos agora calcular uma expressão para a projeção de um vetor arbitrário c sobre o espaço nulo de A.

Teorema: Os vetores $\hat{c}$ e $\bar{c}$ satisfazem:

$$\hat{c} = P_A c, \qquad \text{onde } P_A = I - A^t(AA^t)^{-1}A,$$

e

$$\bar{c} = A^t(AA^t)^{-1}Ac$$

Demonstração:

Sabemos que c pode ser decomposto em:

$$c = \hat{c} + \bar{c}.$$

Como $\bar{c} \in I(A^t)$, então existe $y \in R^m$ tal que $\bar{c} = A^t y$, logo:

$c = \hat{c} + A^t y$. Pré-multiplicando por A, temos:

$A\,c = A\,\hat{c} + AA^t y$.

Como $A\,\hat{c} = 0$ [pois $\hat{c} \in N(A)$], então $AA^t y = A\,c$,

e como AA^t é não singular, temos:

$$y = (AA^t)^{-1}A\,c$$

Finalmente, substituindo a expressão acima em:

$\bar{c} = A^t y$ e $\hat{c} = c - \bar{c}$, obtemos:

$$\bar{c} = A^t(AA^t)^{-1}A\,c \quad e \quad \hat{c} = P_A c,$$

onde $P_A = I - A^t(AA^t)^{-1}A$.

Completando a demonstração.

A matriz P_A é a *matriz de projeção* sobre o espaço nulo de A. O cálculo de P_A inclui uma inversão de matriz, e será sempre o procedimento mais trabalhoso em todos os algoritmos de pontos interiores que estudaremos.

4.2. MÉTODO AFIM PRIMAL CANALIZADO [46].

Considere o Problema de fluxo de custo mínimo formulado a seguir:

$$(PA)\quad \begin{array}{ll} \text{Minimizar} & \begin{bmatrix} c^t, & 0 \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} \\ \text{Sujeito a} & \begin{bmatrix} A & 0 \\ I & I \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix} \\ & \begin{bmatrix} x \\ s \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{array}$$

O dual de (PA) é:

$$(DA)\quad \begin{array}{ll} \text{Maximizar} & \begin{bmatrix} b^t, & d^t \end{bmatrix} \begin{bmatrix} y \\ v \end{bmatrix} \\ \text{Sujeito a} & \begin{bmatrix} A^t & I \\ 0 & I \end{bmatrix} \begin{bmatrix} y \\ v \end{bmatrix} \leq \begin{bmatrix} c \\ 0 \end{bmatrix} \\ & \begin{bmatrix} y \\ v \end{bmatrix} \quad \text{Irrestrito} \end{array}$$

e as condições de folgas complementares podem ser escritas como:

$$\begin{bmatrix} X & 0 \\ 0 & S \end{bmatrix} \left(\begin{bmatrix} c \\ 0 \end{bmatrix} - \begin{bmatrix} A^t & I \\ 0 & I \end{bmatrix} \begin{bmatrix} y \\ v \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

onde $X = diag(x)$ e $S = diag(s)$.

Dado um ponto $\begin{bmatrix} x \\ s \end{bmatrix}$ satisfazendo $Ax = b$ e $x + s = d$, pode-se obter uma estimativa das variáveis duais $\begin{bmatrix} y \\ v \end{bmatrix}$ resolvendo o problema:

$$\text{Min } F(y,v) = \begin{bmatrix} Xc - XA^t y - Xv \\ - Sv \end{bmatrix}^t \begin{bmatrix} Xc - XA^t y - Xv \\ - Sv \end{bmatrix}$$

onde $X = \text{diag}(x)$ e $S = \text{diag}(s)$. Observe que a função $F: R^{m+n} \longrightarrow R$ é quadrática, convexa e não negativa. Para resolver este problema, diferenciamos F com relação a **y** e **v**:

$$\nabla_y F = 2\left[-AX^2c + AX^2A^t y + AX^2 v \right]$$

$$\nabla_v F = 2\left[-X^2c + X^2A^t y + (X^2 + S^2)v \right]$$

Fazendo:

$$\begin{cases} \nabla_y F = 0 \quad \Rightarrow \quad (AX^2A^t)y + (AX^2)v = AX^2c & (1) \\ \nabla_v F = 0 \quad \Rightarrow \quad (X^2A^t)y + (X^2+S^2)v = X^2c & (2) \end{cases}$$

Pré-multiplicando a relação (2) pela expressão:

$-(AX^2)(X^2+S^2)^{-1}$, temos:

$$\begin{cases} (AX^2A^t)y + (AX^2)v = AX^2c \\ -AX^2(X^2+S^2)^{-1}(X^2A^t)y - AX^2v = -AX^2(X^2+S^2)^{-1}X^2c \end{cases}$$

Somando ambos os membros, obtemos:

$$A\left[X^2 - X^2(X^2+S^2)^{-1}X^2 \right]A^t y = A\left[X^2 - X^2(X^2+S^2)^{-1}X^2 \right]c$$

Mas:

$$X^2 - X^2(X^2+S^2)^{-1}X^2 = X^2\left[I - (X^2+S^2)^{-1}X^2 \right]$$

$$= X^2\left[(X^2+S^2)^{-1}(X^2+S^2) - (X^2+S^2)^{-1}X^2\right]$$

$$= X^2(X^2+S^2)^{-1}\left[X^2 + S^2 - X^2 \right]$$

Portanto:

$$X^2 - X^2(X^2+S^2)^{-1}X^2 = X^2(X^2+S^2)^{-1}S^2$$

Fazendo: $\hat{X} = \text{diag}(\hat{x}_i^2)$ (3)

onde $$\hat{x}_j = \frac{x_j \; s_j}{\left(x_j^2 + s_j^2 \right)^{1/2}}$$

Temos:

$$(A \hat{X} A^t) y = A \hat{X} c$$

$$\therefore \quad y = (A\hat{X}A^t)^{-1}A\hat{X} c$$

Em resumo, dado um ponto inicial, $\begin{bmatrix} x \\ s \end{bmatrix}$ satisfazendo $x + s = d$, defina:

- Variáveis duais (preços) $\quad y = \left(A\hat{X}A^t\right)^{-1} A \hat{X} c$

- Custos relativos $\bar{c} = c - A^t y$

- Direção de movimento $\Delta x = -\hat{X}\,\bar{c}$ onde $\hat{X}$ é dado como em (3)

MÉTODO AFIM PRIMAL CANALIZADO.

Seja x um ponto inicial *interior factível* e $\alpha \in (0,1)$.

$k \longleftarrow 0$

Repita

$s \longleftarrow d - x$

$\hat{X} \longleftarrow diag(\hat{x}_i^2)$

$y \longleftarrow (A\hat{X}A^t)^{-1}A\hat{X}\,c$

$\bar{c} \longleftarrow c - A^t y$

$\Delta x \longleftarrow -\hat{X}\,\bar{c}$

$\delta \longleftarrow \text{Min}\left\{ \text{máx}\,[\,-x_i/\Delta x_i,\ (d_i - x_i)/\Delta x_i\,] \right\}$

$x \longleftarrow x + \alpha\,\delta\,\Delta x$

$k \longleftarrow k + 1$

Até convergir.

Onde:

$$\hat{x}_i = \frac{x_i\ s_i}{\left[x_i^2 + s_i^2 \right]^{1/2}}$$

Um método mais simples [46] para calcular os $\hat{x}_i$ é fazer:

$$\hat{x}_i \longleftarrow \text{Min}\ [\ x_i\ ,\ s_i\ ]$$

Os pontos importantes a serem discutidos posteriormente são:

- a resolução de $(\mathbf{A}\hat{\mathbf{X}}\mathbf{A}^t)y = \mathbf{A}\hat{\mathbf{X}}c$ [vide seção 5.5]
- a escolha do ponto inicial *interior factível*;
- o critério de convergência;
- a escolha de α;

Escolha do ponto inicial interior factível.

Neste caso faremos um procedimento análogo ao do método Duas fases **(Simplex)**:

Fase I **[46]**

Para encontrar um ponto inicial interior factível para o problema **(PA)**, consideremos o problema:

$$\text{Minimizar} \qquad \lambda$$

$$\textbf{(P1)} \qquad \text{Sujeito a} \qquad [A\ \rho]\begin{bmatrix} x \\ \lambda \end{bmatrix} = b$$

$$0 \leq x \leq d$$

Onde $\rho = b - Ax^0$ e x^0 é tal que $0 < x^0 < d$

Observações:

- A variável λ é uma variável irrestrita [$\lambda \in \mathbf{R}$].
- O ponto $(x,\lambda) = (x^0,1)$ é factível para **(P1)**.
- Qualquer ponto factível para (P1) que satisfaz $\lambda=0$, também satisfaz $Ax = b$.

- Se o valor mínimo de λ é positivo, então **(PA)** é infactível.

As *observações* acima explicam porque nós queremos minimizar λ.

Se podemos encontrar um par factível (x,λ) com $\lambda < 0$, então podemos tomar uma Combinação Linear deste ponto com $(x^0,1)$ para encontrar um novo ponto que tenha $\lambda=0$. Isto nos dá uma forma de obter um ponto inicial interior factível para o Problema **(PA)**.

Se $\mathbf{x}$ e $\mathbf{y}$ são vetores de $\mathbf{R}^n$, denotaremos por $\mathbf{x} \wedge \mathbf{y}$ o vetor com componentes iguais ao mínimo de suas correspondentes componentes de $\mathbf{x}$ e $\mathbf{y}$, isto é, $(x \wedge y)_i = \min (x_i, y_i)$ $i = 1, \ldots, n$. Análogamente $\mathbf{x} \vee \mathbf{y}$ denota as componentes máximas, é dizer, $(x \vee y)_i = \max (x_i, y_i)$.

Em resumo, dado um ponto x que satisfaz $0 < x < d$, o algoritmo é:

Repita

$s \longleftarrow d - x$

$\rho \longleftarrow b - Ax$

$z \longleftarrow - \hat{X}^2 A^t B \rho$ onde $B = (A\hat{X}^2A^t)^{-1}$

$\gamma \longleftarrow \text{Max} (z/x \wedge -z/s)$

$\delta \longleftarrow \text{Min} (z/x^2 \vee -z/s^2)$

$M \longleftarrow \text{Max} (x \wedge s)$

se $\gamma + M\delta < \theta\varepsilon/n$ onde $\theta = \rho^t B\rho$

então pare [problema infactível]

senão se $\gamma < \beta$

então (x <—— x - z; vá para fase II)

senão x <—— x - $(\beta/\gamma)z$

Critério de Convergência para o Problema Primal Canalizado (PA). **[1]**

Dado ε uma tolerância prefixada:

Na k-ésima iteração [k:inteiro], testar:

Se

$$\frac{\left| c^t x^{(k)} - c^t x^{(k-1)} \right|}{\text{Max}\left[1, \left| c^t x^{(k-1)} \right| \right]} < \varepsilon$$

Então Pare [$x^{(k)}$ é uma solução ε-ótima].

Onde: $\varepsilon = 10^{-\theta}$.

Escolha do Parâmetro α. **[1]**

Como $\alpha \in (0,1)$, com isto nós garantimos que o próximo ponto seja ainda interior, então pode-se trabalhar com os seguintes valores:

α = 0.90 ou 0.95 ou 0.99; em todas as iterações.

4.3. MÉTODO AFIM DUAL CANALIZADO [46].

O problema de fluxo de custo mínimo pode ser escrito na seguinte forma:

$$\text{Minimizar} \qquad [c^t,\ 0]\begin{bmatrix} x \\ s \end{bmatrix}$$

$$(PA') \qquad \text{Sujeito a} \qquad \begin{bmatrix} A & 0 \\ I & I \end{bmatrix}\begin{bmatrix} x \\ s \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$

$$\begin{bmatrix} x \\ s \end{bmatrix} \geq 0$$

cujo problema dual é:

$$\text{Maximizar} \qquad [\ b^t,\ -u^t,\ 0\ ]\begin{bmatrix} y \\ v \\ w \end{bmatrix}$$

$$(DA') \qquad \text{Sujeito a} \qquad [\ A^t \quad -I \quad I\ ]\begin{bmatrix} y \\ v \\ w \end{bmatrix} = c$$

$$\begin{bmatrix} v \\ w \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

e cujas condições de folgas complementares são

$W\ x = 0$ onde $W = \text{diag}(w)$

$V\ s = 0$ onde $V = \text{diag}(v)$

Dado um ponto $\begin{bmatrix} y \\ v \\ w \end{bmatrix}$, com $A^t y - v + w = c$. Uma estimativa das variáveis primais $\begin{bmatrix} x \\ s \end{bmatrix}$, pode ser obtida resolvendo o problema:

$$\text{Minimizar} \qquad G(x,s) = \begin{pmatrix} W & x \\ V & s \end{pmatrix}^t \begin{pmatrix} W & x \\ V & s \end{pmatrix}$$

Sujeito a $\quad A\,x = b$

$$x + s = d$$

Onde : $\quad W = \mathrm{diag}(w) \quad e \quad V = \mathrm{diag}(v)$

A função $G: R^{n+n} \longrightarrow R$ é quadrática, convexa e não negativa.

Para resolver este problema utilizamos o lagrangeano

$$L(x,s,\lambda_1,\lambda_2) = G(x,s) + \lambda_1^t(Ax - b) + \lambda_2^t(x + s - d)$$

Aplicando as condições de *KUHN TUCKER*, temos:

$$\nabla_x L(x,s,\lambda_1,\lambda_2) = 0 \quad \Rightarrow \quad \nabla_x G(x,s) + A^t\lambda_1 + \lambda_2 = 0$$

$$\nabla_s L(x,s,\lambda_1,\lambda_2) = 0 \quad \Rightarrow \quad \nabla_s G(x,s) + \lambda_2 = 0$$

$$\nabla_{\lambda_1} L(x,s,\lambda_1,\lambda_2) = 0 \quad \Rightarrow \quad A\,x - b = 0$$

$$\nabla_{\lambda_2} L(x,s,\lambda_1,\lambda_2) = 0 \quad \Rightarrow \quad x + s - d = 0$$

O que implica:

$$2\,W^2\,x + A^t\lambda_1 + \lambda_2 = 0 \qquad (1)$$

$$2\,V^2\,s + \lambda_2 = 0 \qquad (2)$$

$$A\,x - b = 0 \qquad (3)$$

$$x + s - d = 0 \qquad (4)$$

De (2): $\quad \lambda_2 = -2\,V^2\,s$

Substituindo a expressão acima na relação (1), temos:

$$2\,W^2\,x + A^t\lambda_1 - 2\,V^2 s = 0$$

ou $\quad W^2\,x - V^2\,s + \frac{1}{2}A^t\lambda_1 = 0 \qquad (5)$

Agora, pré-multiplicando a relação (4) por V^2, obtemos:

$$V^2 x + V^2 s - V^2 d = 0 \qquad (6)$$

Somando (5) e (6), obtemos:

$$(W^2 + V^2) x + \frac{1}{2} A^t \lambda_1 = V^2 d$$

então:

$$x = \left(W^2 + V^2 \right)^{-1} \left[V^2 d - \frac{1}{2} A^t \lambda_1 \right]$$

Substituindo x na relação (3), obtemos:

$$A \left(W^2 + V^2 \right)^{-1} \left[V^2 d - \frac{1}{2} A^t \lambda_1 \right] = b$$

Portanto:

$$\lambda_1 = -2 \left[A (W^2 + V^2)^{-1} A^t \right]^{-1} \left[b - A (W^2 + V^2)^{-1} V^2 d \right]$$

Fazendo $\hat{Y} = (W^2 + V^2)^{-1}$, obtemos:

$$\lambda_1 = -2 \left[A \hat{Y} A^t \right]^{-1} \left[b - A \hat{Y} V^2 d \right]$$

e $$x = \hat{Y} \left\{ V^2 d + A^t (A \hat{Y} A^t)^{-1} \left[b - A \hat{Y} V^2 d \right] \right\}$$

Onde: $\hat{Y} = \text{diag} \left(\frac{1}{v_i^2 + w_i^2} \right)$ $i = 1, \ldots, n$

Em resumo, dado um ponto $\begin{bmatrix} y \\ v \\ w \end{bmatrix}$ tal que $A^t y - v + w = c$,

defina:

- Variáveis de folga: $w = c - A^t y + v$ $\quad [\ \bar{c} = c - A^t y = w - v\ ]$

- Direção de movimento: $\left(\Delta y, \Delta v, \Delta w\right)$ dado por:

$$\Delta y = (A\hat{Y}A^t)^{-1}\left[\ b - A\hat{Y}V^2\ d\ \right] \qquad (7)$$

$$\Delta v = V^2(d - x), \text{ onde } x := \hat{Y}\ (\ V^2 d + A^t \Delta y\ )$$

$$\Delta w = -W^2\ x$$

- Variáveis primais: $x = \hat{Y}(\ V^2\ d + A^t\ \Delta y\ )$,

onde Δy está dado como em (7), e $s = d - x$.

METODO AFIM DUAL CANALIZADO.

Seja $\begin{bmatrix} y \\ v \\ w \end{bmatrix}$ um ponto inicial *interior factível*, satisfazendo $A^t y - v + w = c$, e $\alpha \in (0,1)$.

$k \leftarrow 0$

Repita

$w \leftarrow c - A^t y + v$

$\hat{Y} \leftarrow \mathrm{diag}\left(\ 1\ /\ (\ v_i^2 + w_i^2\ )\ \right)$

$\Delta y \leftarrow (A\hat{Y}A^t)^{-1}\left[\ b\ -\ A\hat{Y}V^2\ d\ \right]$ $\quad [\ V=\mathrm{diag}(v)\ ]$

$x \leftarrow\!\!- \hat{Y}(V^2 d + A^t \Delta y)$

$\Delta v \leftarrow\!\!- -V^2(d - x)$

$\Delta w \leftarrow\!\!- -W^2 x$ [W=diag(w)]

$\delta \leftarrow\!\!- \text{Min}\left[\text{Min}(-w_j / \Delta w_j , -v_j / \Delta v_j) \right]$

$y \leftarrow\!\!- y + \alpha \delta \Delta y$

$v \leftarrow\!\!- v + \alpha \delta \Delta v$

$k \leftarrow\!\!- k + 1$

Até convergir

Os pontos importantes a serem discutidos são:

- a determinação de Δy [vide seção 5.5]
- a escolha do ponto inicial *interior factível*
- o critério de convergência
- a escolha de α

Inicialização do Método Afim Dual Canalizado.

Neste caso, nós não precisamos da fase I, ou seja, a inicialização no Dual canalizado é da seguinte forma:

Dado $y \in R^m$, considere $k > 0$ [$k \in R$]. Logo temos:

Se $(c - A^t y)_i \geq 0$,

então $v_i \leftarrow\!\!- k$, e $w_i \leftarrow\!\!- (c - A^t y)_i + v_i > 0$;

Caso contrario

$w_i \leftarrow\!\!- k$, e $v_i \leftarrow\!\!- -(c - A^t y)_i + w_i > 0$

Portanto temos um ponto (y,v,w) interior factível, isto é, satisfazendo:

$$A^t y - v + w = c \ , \ v > 0 \ e \ w > 0.$$

Observação:

Uma maneira mais simples é:

Faça $y = 0$ e considere $k > 0$ [$k \in R$]. Logo temos:

Se $c_i \geq 0$,

então $v_i \leftarrow k$, e $w_i = c_i + v_i > 0$

Caso contrário

$$w_i \leftarrow k, \text{ e } v_i = -c_i + w_i > 0.$$

Portanto o ponto (y,v,w) é interior factível.

Critério de Convergência para o Método Afim Dual Canalizado.

Dada ε uma tolerância prefixada:

Na k-ésima iteração [k:inteiro], testar:

Se

$$\frac{| b^t y^{(k)} - u^t v^{(k)} - b^t y^{(k-1)} + u^t v^{(k-1)} |}{\text{Max}\ [\ 1, | b^t y^{(k-1)} - u^t v^{(k-1)} |\ ]} < \varepsilon$$

Então Pare [$y^{(k)}$, $v^{(k)}$ é uma solução ε-ótima], onde $\varepsilon = 10^{-8}$.

Escolha do Parâmetro α.

Como $\alpha \in (0,1)$, pois com isto nós garantimos que o próximo ponto seja ainda interior, então pode-se trabalhar com os seguintes valores:

$\alpha = 0.99$ às 10 primeiras iterações e $\alpha = 0.95$ às iterações seguintes

CAPITULO V

RESOLUÇÃO DE SISTEMAS DE EQUAÇÕES LINEARES

5.1. INTRODUÇÃO.

Em todos os métodos apresentados nos **capítulos III e IV** (vide seções 3.2, 3.3, 4.2, 4.3), temos que o maior esforço computacional está na resolução de um sistema de equações lineares de ordem n (n é o número de linhas da matriz de posto completo A), isto é, temos os seguintes sistemas:

Para o Problema (P):

$$B\, x_B = b$$

Para o Problema (D):

$$B^t\, y = c_B$$

Para o Problema (PA):

$$(A\hat{X}A^t)\, y = A\hat{X}\, c$$

Para o Problema (DA):

$$(A\hat{Y}A^t)\Delta y = b - A\,\hat{Y}\,V^2\,u$$

Onde **B** é uma matriz não singular e $(A\hat{X}A^t)$ e $(A\hat{Y}A^t)$ são matrizes simétricas e definidas positivas. Portanto a complexidade computacional dos métodos simplex e de pontos interiores considerados depende basicamente da resolução de sistemas do tipo

$$Q\, z = q$$

onde **Q** é simétrica e definida positiva para os problemas **PA** e **DA**, e onde **Q** é não singular para os problemas **P** e **D**.

Para resolver $Q\ z = q$ podem ser utilizados métodos diretos (fatoração **LU**, fatoração Cholesky LL^T, fatoração LDL^T) e métodos iterativos (gradiente conjugado, Gauss-Seidel, sobre-relaxação sucessiva). A seguir, discutiremos alguns

destes métodos.

5.2. MÉTODOS DIRETOS [15].

Métodos diretos são aqueles que, a menos de erros de arrendondamento, fornecem a solução exata do sistema $Q\ z = q$, após um número finito de iterações.

5.2.1. Sistemas Triangulares [15].

Requer que a matriz Q seja triangular não singular. A resolução de um sistema triangular é realizado com eficiência, pelo método " backward substitution ". O procedimento abaixo, resolve o sistema $Q\ z = q$. Suponha que $Q_{ij} = 0$ para $i<j$.

```
Procedure Resolve Sistema Triangular ( n, Q, q, z )
i, j : integer;
BEGIN
```

$$z_n := q_n \ / \ Q_{nn}$$

for i:=n-1 **downto** 1 **do**

$$z_i = \frac{q_i - \sum_{j=i+1}^{n} Q_{ij} z_j}{Q_{ii}}$$

END.

Analogamente, se resolve um sistema triangular inferior, neste caso usa-se o procedimento " forward substitution ".

5.2.2. Fatoração **LU** [15].

Requer que a matriz **Q** seja não singular. Neste caso a matriz **Q**, é decomposta na forma **PQ = LU**, onde **P** é uma matriz de permutação, **L** é uma matriz triangular inferior com 1's na diagonal e **U** é uma matriz triangular superior.

É importante notar que, as matrizes **L** e **U** podem ser armazenadas na própria matriz **Q**, pois a matriz **L** tem 1's na diagonal.

Desta forma, **Q z = q** pode ser escrito como:

PQ z = P q <===> **(L U)z = P q,**

e a solução **z** pode ser obtida calculando **w** que satisfaça:

L w = P q [*Sistema triangular inferior*]

e a seguir resolvendo

U z = w [*Sistema triangular superior*]

Ou seja, resolvendo dois sistemas triangulares.

5.2.3. FATORAÇÃO CHOLESKY $L L^t$ [41].

Requer que a matriz Q em Q z = q seja simétrica e positiva definida. Consiste em determinar uma matriz triangular inferior L tal que $Q = L L^t$. Os elementos da matriz $L = (L_{ij})$ podem ser calculados pela fórmula recursiva:

$$L_{jj} = \left[Q_{jj} - \sum_{k=1}^{j-1} L_{jk}^2 \right]^{1/2} \qquad j=1,..,n$$

$$L_{ij} = \frac{1}{L_{jj}} \left[Q_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk} \right] \qquad i>j, \qquad j=1,..,n$$

Desta forma, Q z = q pode ser escrito como:

$$(L L^t) z = q,$$

e a solução z pode ser obtida calculando w que satisfaça:

$$L w = q \qquad [Sistema\ triangular\ inferior]$$

e a seguir resolvendo

$$L^t z = w \qquad [Sistema\ triangular\ superior]$$

Ou seja, resolvendo dois sistemas triangulares.

O Procedimento **CHOLESKY** abaixo obtem a matriz triangular inferior **L**, tal que $LL^t = Q$:

```
Procedure  CHOLESKY ( n, Q, L )
i, j, k     :  integer;
S           :  real;
BEGIN
     L11 := sqrt(Q11);
```

```
for j:=2 to n do L_j1 := Q_j1 / L_11;
for j:=2 to n do
begin
    S:=0;
    for k:=1 to (j-1) do S:=S + L_jk^2;
    L_jj := sqrt(Q_jj - S);
    for  i:=j+1 to n do
    begin
        S:=0;
        for k:=1 to (j-1) do S:=S + L_ik * L_jk;
        L_ij := (Q_ij - S)/L_jj
    end
end
END
```

5.2.4. Fatoração Cholesky LDL^t [33] [41].

Requer que a matriz Q seja simétrica e positiva definida. Neste caso a matriz Q, é decomposta na forma $Q = LDL^t$, onde L é uma matriz triangular inferior com 1's na diagonal e D é uma matriz diagonal. Neste tipo de decomposição, não é necessário extrair raiz quadrada, pois em algumas máquinas, o tempo requerido por esta operação é três operações de soma.

O procedimento abaixo obtem as matrizes L e D, tal que $LDL^t = Q$.

```
Procedure CHOLESKY2 ( n, Q, L, D )
i,j,k : integer;

BEGIN
      for i:=1 to n do
      begin
            D_ii := Q_ii
            for j:=i+1 to n do
            begin
                  L_ji := Q_ji / Q_ii
                  for k:=j to n do Q_kj := Q_kj - L_ji*Q_ki
            end
      end
END
```

5.3. MÉTODOS ITERATIVOS.

Existe uma outra família de métodos que fazem uso apenas dos elementos da matriz Q original. Estes métodos consistem de algoritmos simples para obter a partir de um vetor $z^{(k)}$ um outro vetor, $z^{(k+1)}$, que depende também de Q e q. Ou seja, a matriz Q é mantida sem alterações. Estes métodos pertencem à classe dos métodos iterativos para resolver um sistema Linear.

5.3.1. MÉTODO GRADIENTE CONJUGADO (GC) [LUENBERGER [35]]

Resolver Q z = q é equivalente a resolver o problema:

Minimizar $F(z) = \frac{1}{2} z^t Q\, z - z^t q$ (PQ)

onde Q é simétrica e definida positiva. Observe que F é uma função quadrática e convexa, e que tem um único ponto de mínimo z^*, isto é:

$\nabla F(z) = 0$ implica $z^* = Q^{-1} q$ [solução de Q z = q]

A seguir daremos algumas definições e teoremas, os quais serão usados posteriormente:

Definição 1:

Seja $Q \in R^{n \times n}$ uma matriz simétrica.

Um conjunto finito de vetores $d_0, d_1, \ldots, d_k$ é um conjunto Q-conjugado (Q-ortogonal), se:

$$d_i^t\, Q\, d_j = 0 \quad \forall\, i \neq j$$

Teorema 1.

Se Q é definida positiva e o conjunto de vetores não nulos $d_0, d_1, \ldots, d_k$ é Q-conjugado, então $\{d_0, d_1, \ldots, d_k\}$ é

linearmente independente.

Demonstração:

Suponha que existem constantes α_i, $i=0,1,\ldots,k$ tais que:

$$\alpha_0 \mathbf{d}_0 + \ldots + \alpha_k \mathbf{d}_k = \mathbf{0}$$

Pré-multiplicado a expressão acima por $\mathbf{d}_i^t Q$, obtemos:

$$\alpha_i \mathbf{d}_i^t Q \mathbf{d}_i = 0$$

Tendo em vista que $\mathbf{d}_i^t Q \mathbf{d}_i > 0$, pois Q é definida positiva, então temos que:

$$\alpha_i = 0$$

Portanto $\{\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_k\}$ é linearmente independente.

Vejamos agora, porque a noção de Q-conjugado é útil para a solução do problema (PQ).

Associado à matriz Q, temos o conjunto de n vetores não nulos $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{n-1}$ Q-conjugados. Pelo teorema 1 nós sabemos que o conjunto $\{\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{n-1}\}$ é linearmente independente, o que implica que a solução z^* de (PQ) ou $Q z = q$, pode ser escrita assim:

$$z^* = \alpha_0 \mathbf{d}_0 + \ldots + \alpha_{n-1} \mathbf{d}_{n-1}$$

para algum conjunto de α_i's. De fato, pré-multiplicando-a expressão acima por $\mathbf{d}_i^t Q$, obtemos:

$$\mathbf{d}_i^t Q z^* = \alpha_i \mathbf{d}_i^t Q \mathbf{d}_i$$

→

$$\alpha_i = \frac{\mathbf{d}_i^t Q z^*}{\mathbf{d}_i^t Q \mathbf{d}_i} = \frac{\mathbf{d}_i^t q}{\mathbf{d}_i^t Q \mathbf{d}_i}$$

Isto mostra que os α_i's e consequentemente a solução z^* pode ser encontrada pela simples avaliação de produtos

escalares. Portanto o resultado final é:

$$z^* = \sum_{i=0}^{n-1} \frac{d_i^t\, q}{d_i^t\, Q\, d_i}\, d_i$$

Teorema 2 (teorema de direções conjugadas).

Seja $\left\{ d_i \right\}_{i=0}^{n-1}$ um conjunto de vetores não nulos Q-ortogonais. $\forall\ z_o \in R^n$ a sequência $\left\{ z_k \right\}$ gerada segundo:

$$z_{k+1} = z_k + \alpha_k d_k \qquad k \geq 0 \qquad (1)$$

com $$\alpha_k = - g_k^t\, d_k \,/\, d_k\, Q\, d_k \qquad (2)$$

e $$g_k = Q\, z_k - q \qquad (3)$$

converge à solução única z^*, de $Qz = q$, após n passos, isto é, $z_n = z^*$.

Demonstração : [vide referência 35 **].**

Descrição do Método.

O método gradiente conjugado é um método de direções conjugadas que é obtido selecionando os vetores de direção como uma versão conjugada dos sucessivos gradientes. Assim as direções não são especificadas de antemão, sendo determinadas sequencialmente a cada passo de uma determinada iteração. No passo k é avaliado o vetor gradiente (negativo) e é adicionado a ele uma combinação linear dos anteriores vetores direção para obter um novo vetor direção Q-conjugado.

Existem três vantagens fundamentais neste método:

1. A menos que a solução seja atingida em menos do que n passos, o gradiente é sempre não nulo e linearmente independente de todos os vetores de direção anteriores. Na verdade, o gradiente g_k é ortogonal ao sub-espaço gerado por $d_0, d_1, \ldots, d_{k-1}$. Se a solução é alcançada antes de n passos, o gradiente desaparece e o processo termina, quer dizer, sua existência é desnecessária, neste caso, para encontrar direções adicionais.

2. A vantagem mais importante do método gradiente conjugado é a fórmula simples que é usada para determinar o novo vetor direção.

3. Posto que as direções são fundamentadas no gradiente, o processo avança uniformemente na direção da solução a cada passo.

Método gradiente conjugado.

Dada a matriz $Q \in R^{nxn}$ simétrica e definida positiva e um vetor $z_0 \in R^n$ (qualquer).

Faça $g_0 := Qz_0 - q$; $d_0 := - g_0$

Para $k = 0,1,\ldots$ faça:

$$\alpha_k \longleftarrow - d_k^t g_k \,/\, d_k^t Q d_k \qquad (4)$$

$$z_{k+1} \longleftarrow z_k + \alpha_k d_k \qquad (5)$$

$$\beta_k \longleftarrow g_{k+1}^t Q d_k \,/\, d_k^t Q d_k \qquad (6)$$

$$d_{k+1} \longleftarrow -g_{k+1} + \beta_k d_k \qquad (7)$$

onde $g_k = Qz_k - q$.

Neste método o primeiro passo é idêntico a um passo do método gradiente, isto é, caminho na direção oposta do gradiente; cada passo seguinte move-se em uma direção que é uma combinação linear do vetor gradiente atual e o vetor direção anterior.

A característica atraente do método gradiente conjugado é a fórmula simples (6) e (7), para atualização do vetor direção.

Verificação do Método.

Para verificar que o método gradiente conjugado é um método de direções conjugadas, é necessário verificar que os vetores $\{ \mathbf{d}_k \}$ são Q-conjugados. É fácil provar isso, provando simultaneamente outras propriedades do método. Isto é feito no teorema abaixo onde a notação $[\mathbf{d}_0, \ldots, \mathbf{d}_k]$ é usada para denotar o sub-espaço gerado pelos vetores $\mathbf{d}_0, \ldots, \mathbf{d}_k$.

Teorema 3 (Teorema gradiente conjugado).

O método gradiente conjugado definido pelas relações (4), (5), (6) e (7) é um método de direções conjugadas. Se ele não termina em z_k, então:

(a) $[\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_k] = [\mathbf{g}_0, Q\,\mathbf{g}_0, \ldots, Q^k\,\mathbf{g}_0]$

(b) $[\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_k] = [\mathbf{g}_0, Q\,\mathbf{g}_0, \ldots, Q^k\,\mathbf{g}_0]$

(c) $\mathbf{d}_k^t\, Q\, \mathbf{d}_i = 0$ para $i \leq k-1$

(d) $\alpha_k = \mathbf{g}_k^t\, \mathbf{g}_k \;/\; \mathbf{d}_k^t\, Q\, \mathbf{d}_k$

(e) $\beta_k = \mathbf{g}_{k+1}^t\, \mathbf{g}_{k+1} \;/\; \mathbf{g}_k^t\, \mathbf{g}_k$

Demonstração [vide referência 35]

Observações:

As partes (a) e (b) deste teorema constituem um enunciado formal da interrelação entre os vetores de direção e os

vetores gradiente. A parte (c) é a equação que verifica que o método gradiente conjugado é um método de direções conjugadas. As partes (d) e (e) são identidades que fornecem fórmulas alternativas para α_k e β_k, as quais são frequentemente mais convenientes que as fórmulas originais.

Cada iteração do algoritmo precisa $n^2 + O(n)$ multiplicações e divisões, de tal forma que a solução de $Q\ z = q$ requer $n^3 + O(n^2)$ operações, pois, na verdade a solucão é encontrada em n passos (no máximo), ou seja este é um método finito.

O método GC pode convergir em menos de n iteracões.

5.3.2. MÉTODO GAUSS SEIDEL (GS) [22].

Em geral, os métodos iterativos determinam uma sequência de pontos z^1, z^2, ... satisfazendo:

$$z^{k+1} = M\ z^k + d$$

Considere que a matriz Q é expressa por:

$$Q = L + D + L^t$$

onde L é triangular inferior com zeros na diagonal e D é uma matriz diagonal. Se z^{k+1} é calculado a partir de z^k de modo a satisfazer:

$$(L + D)\ z^{k+1} + L^t\ z^k = q$$

então

$$z^{k+1} = -(D + L)^{-1}\ L^t\ z^k + (D + L)^{-1}\ q$$

Teorema: Se Q é simétrica e definida positiva, então o método Gauss-Seidel converge independentemente do vetor inicial.

Prova [Vide referência **41].**

5.3.3. MÉTODO SOBRE-RELAXAÇÃO SUCESSIVA (SOR) [16].

Para resolver Q z = q podemos usar um método iterativo mais geral que o método Gauss-Seidel, que depende de um parámetro ω (ω número real), definido assim:

$$(D - \omega L)z^{k+1} = \left[(1 - \omega)D + \omega L^{t}\right]z^{k} + \omega\, q$$

onde D é uma matriz diagonal e L é uma matriz triangular inferior.

Para ω=1 este método coincide com o método GS. Estritamente falando, o termo " sobre-relaxação " é usado somente com $\omega > 1$. Claramente a expressão acima é equivalente a:

$$z^{k+1} = (D-\omega L)^{-1}\left[(1-\omega)D + \omega\, L^{t}\right]z^{k} + (D - \omega L)^{-1}\omega\, q$$

Teorema: Se Q é simétrica e definida positiva, então para qualquer ω, $0 < \omega < 2$, o método **SOR** converge independentemente do vetor inicial.

Prova [Vide referência **16].**

5.4. ESTRUTURA DE DADOS PARA PROGRAMAÇÃO EM REDES.

Uma estrutura de dados adequada para o uso de métodos iterativos é apresentada neste trabalho. Considere o problema :

$$\begin{array}{lll} & \text{Minimizar} & c^t x \\ (P) & \text{Sujeito a} & A\,x = b \\ & & 0 \le x \le d \end{array}$$

onde a matriz **A** é a matriz de incidência nóxarco associada ao grafo **G** = [*N*,*A*], **b** é o vetor de demandas associado aos nós da rede **G**, **c** é o vetor de custos associado aos arcos da rede **G**, e o vetor **d** é o vetor de capacidades, isto é, a quantidade máxima de fluxo que pode passar por cada arco na rede; o nosso objetivo é, encontrar um vetor de fluxos **x**, solução ótima de (P). Observe que a matriz **A** tem posto (n-1) e que **I** [**A**|**b**] = 0.

A estrutura de dados é formada apenas por vetores unidimensionais com m ou n elementos, onde n é o número de nós e m é o número de arcos. Há dois vetores que identificam os nós de cada arco da rede, isto é: (t_j, h_j) indica a cauda (*'tail '*) e a cabeça (*'head '*) do arco $j \in A$. Como há n nós e m arcos, então tail e head são vetores com dimensão m (não dependem do número de nós).

A estrutura de dados em questão utiliza os seguintes vetores:

b [1..n] : demandas nos nós

c [1..m] : custos nos arcos

d [1..m] : capacidade dos arcos

x [1..m] : fluxos nos arcos

y [1..n] : preços nos nós

t [1..m] : caudas dos arcos

h [1..m] : cabeças dos arcos

Agora daremos algumas definições, que serão usadas

posteriormente: O *Grau externo de um nó* é a quantidade de arcos cuja cauda é este nó. O *Grau interno de um nó* é a quantidade de arcos cuja cabeça é este nó. Assim, também fazem parte da estrutura de dados os vetores:

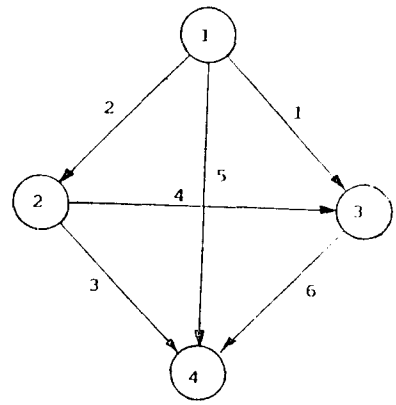
p [1..n] : Grau externo acumulado dos nós

q [1..n] : Grau interno acumulado dos nós

r [1..m] : Lista de arcos ordenados por cauda .

s [1..m] : Lista de arcos ordenados por cabeça.

Assim por exemplo, se tivermos uma rede de 4 nós e 6 arcos, definida por:



Logo, os vetores t, h, p, q, r e s são:

t = [1 1 2 2 1 3]

h = [3 2 4 3 4 4]

p = [3 5 6 6]

r = [1 2 5 3 4 6]

q = [0 1 3 6]

s = [2 1 4 3 5 6]

Desta forma, o número de arcos que sai do nó i é $p_i - p_{i-1} + 1$, assim como o número de arcos que chega no nó i é $q_i - q_{i-1} + 1$ e seus índices são $r[p_{i-1}+1] \ldots r[p_i]$

e $s[q_{i-1}+1]$ .. $s[q_i]$, respectivamente.

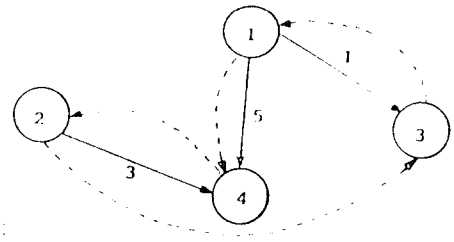
As Bases utilizadas no método simplex são árvores geradoras com raiz fixa no nó artificial root = 0 e são mantidos 3 vetores para caracterização da árvore.

w [1..m] : arco que antecede o nó em questão no caminho que parte da raiz até ele.

u [1..m] : próximo nó na lista fio (thread)

v [1..m] : nó anterior na lista fio (reversed thread)

Por exemplo; considere a árvore geradora associada à base $B = [e_1, e_9, e_5]$



Logo os vetores w, u e v são:

w = [0, 3, 1, 5]

u = [4, 3, 1, 2]

v = [3, 4, 2, 1]

O método Simplex primal usa os seguintes vetores: b, c, d, x, y, t, h, w, u, v, além de um vetor auxiliar de n elementos.

O método Simplex dual usa os seguintes vetores b, c, d, x, y, t, h, w, u, v, além de um vetor auxiliar de m elementos.

O método Out-of-Kilter usa os vetores b, c, d, x, y, t, h, p, q, r, s, além de um vetor auxiliar.

Os métodos Afins (primal e dual), PGRAD, DGRAD e PGAUSS, usam b, c, d, x, y, t, h, além de outros vetores necessários para a implementação da decomposição de cholesky ou do método do gradiente conjugado.

5.5. SOLUÇÃO PARA Q x = q quando Q é triangular.

5.5.1. Resolução de $B x_B = b$

Suponha que B, x_N já estão pré-fixados; ou seja, $x_j=0$ ou $x_j=d_j$ para $j \in N$. Desta forma, deseja-se determinar a solução básica primal definida por (B,x_N):

```
a:=b;
for j:=1 to m do
    if j = w[h_j] and j ≠ w[t_j] then
       begin
       a[h_j] := a[h_j] + x_j
       a[t_j] := a[t_j] - x_j
       end
i := v[root]
repeat
     j:= w_i
     if i=h_j    then    x_j := a_i      else      x_j := -a_i;
     i := v_i
until  i = root
```

O método acima pode ser tornado mais eficiente quando é necessário determinar o ciclo formado na árvore geradora com a inclusão do arco arci escolhido como candidato para entrar na base. Vamos apresentar o algoritmo implementado que além de determinar o ciclo, realiza o teste de razão e atualiza o fluxo.

```
     for i:=0 to n do a_i:=0;
     cr := c_j + y_t_j - y_h_j
     if cr < 0 then begin srce:=h[arci]; sink := t[arci] end
               else begin srce:=t[arci]; sink := h[arci] end
     i:=srce; a_i:=1;
```

```
while i≠root do
      begin
      j := w_i
      i := t_j + h_j - i
      a_i := 1
      end
i := sink; arco:=arci; δ:=d_arco;
while a[i] = 0 do
      begin
      j := w_i
      if i=h_j then Δ := d_j - x_j else Δ := x_j
      if δ>Δ then  begin  δ := Δ; arco := j  end
      i := t_j +h_j -i
      end
apex := i
i := srce;
while i≠apex do
      begin
      j:=w_i
      if i=h_j then Δ:=x_j else Δ:=d_j - x_j
      if δ>Δ then begin δ:=Δ; arco:=j end
      i := h_j +t_j -i;
      end
if δ = ∞ then stop: solução ótima ilimitada;
if cr < 0 then x[arci] := x[arci] + δ
          else x[arci] := x[arci] - δ
i:=sink;
while i≠apex do
      begin
      j:=w_i
      if i=h_j then x_j := x_j +δ else x_j :=x_j -δ
      i:=h_j +t_j -i
      end
```

```
j:=srce;
while i≠apex do
      begin
      j:=w_i
      if i=t_j then x_j := x_j-δ else x_j:=x_j+δ
      i:=h_j+t_j-i
      end
```

Desta forma, **apex** denota o nó mais alto (mais próximo da raiz) do ciclo constituído pelos nós t[arci] ... apex, h[arci] ... apex. Observe que o assinalamento $i = h_j + t_j - i$ faz com que a variável i passe a indicar o pai (na árvore) do nó i. Além disto a_i=1 indica o caminho t[arci] .. root.

5.5.2. Resolução de $\mathbf{B}^t y = c_B$

Suponha que **B** já está definida. Deseja-se determinar a solução básica dual definida por **B**.

```
y[root] := 0
i := u[root]
repeat
      j := w_i
      if i = h_j then y_i := y[t_j] + c_j else y_i:=y[h_j] - c_j
      i := u_i
until  i = root
```

O método acima pode ser tornado mais eficiente quando é necessário determinar a partição dos nós formada na árvore geradora com a retirada do arco arco escolhido para deixar a base. Vamos apresentar o algoritmo implementado que além de determinar a partição de nós realiza o teste de razão.

```
if arco = wlh[arco]
then
     begin
     nodl := h[arco]
     if   x[arco] < 0   then γ := x[arco]
                        else γ := x[arco] - d[arco]
     end
else
     begin
     nodl := t[arco]
     if   x[arco] < 0   then γ := -x[arco]
                        else γ := d[arco] - x[arco]
     end
for i:=0 to n do a[i]:=0
no := nodl;
repeat
      a[no] := 1
      no    := u[no]
until a[antc[no]] = 0
δ := ∞
arci := 0
if γ < 0
then begin
      for j:=1 to m do
          if   a[t[j]] ≠ a[h[j]]  and j≠arco
          then
               begin
               Δ := c[j] + y[t[j]] - y[h[j]]
               if   a[h[j]] = 0   and   δ>Δ   and   x[j]=0
                    then  { δ := Δ; arci := j }  else
               if   a[t[j]] = 0   and   δ>-Δ   and   x[j]=d[j]
```

```
                then  ( δ:= -Δ; arci := j  )
            end
        end
    else begin
        for j:=1 to m do
            if a_t_j ≠ a_h_j  and  j≠arco
            then
                begin
                Δ := c_j + y_t_j - y_h_j
                if  a_t_j = 0  and  δ>Δ  and  x_j=0
                    then  ( δ:=Δ; arci:=j )  else
                if  a_h_j = 0  and  δ>-Δ  and  x_j=d_j
                    then  ( δ:=-Δ; arci:=j )
                end
        end

    for i:=1 to m do a_i:=0
    a[nodl] := 1;
    i := u[nodl];
    k := nodl;
    while ( i ≠ root )  and  ( a_k = 1 ) do
        begin
        a_i := 1
        i := u_i
        j := w_i
        k := t_j + h_j - i
        end
```

Desta forma, **nodl** denota o nó mais alto (mais próximo de raiz) da subárvore criada com a eliminação de **arco**. O vetor (a_i) representa a partição de nós com $a_i=0$ se i não pertence à subárvore e $a_i=1$ se i pertence à subárvore de **nodl**. Observe

que o nó i caminha de acordo com o fio (thread) u da árvore, j é o arco antecessor de i e k é o nó pai de i na árvore.

5.5.3. Atualização da árvore geradora T e dos preços y.

Seja cr o custo relativo do arco que entra na base arci, nodl, o nó mais alto da subárvore formada pela retirada de arco, knot o pái de nodl. no e na (nó anterior) são duas variáveis auxiliares para percorrer o ciclo a partir de arci até nodl; os preços dos nós no caminho percorrido são atualizados do mesmo modo que os vetores u, v, w. Enquanto cada nó deste caminho é denotado pela variável frst, a variável last pesquisa o seu último descendente. Todos os descendentes de frst têm o seu preço atualizado também.

```
a:=0
j:=arci
while no≠knot do
      begin
      transfer ( na, no);
      k:=j; j:=w_no ; w_no :=k;
      na := no;
      no := h_j + t_j - no;
      end
```

onde

```
transfer ( na, frst )
begin
      i:=frst
      repeat    y_i := y_i + cr;
                a_i := frst;
                last:= i;
                i := u_i;
```

```
        until  i=root  or  a[t_{w_i} + h_{w_i} - i] ≠first
        u[ v_{frst} ] := u[ last ]
        v[ u_{last} ] := v[ frst ]
        v[ u_{na}   ] := last
        u[ last     ] := u[ na ]
        v[ frst     ] := na
    end
```

5.6. Resolução de $(\hat{A}\hat{X}A^t)y = \hat{A}\hat{X}c$ pelo gradiente conjugado.

Neste caso é utilizado o vetor $\hat{x}[1..m]$ para armazenar os elementos da diagonal de $\hat{X}$. São também utilizados os vetores auxiliares q, d, r, s onde $q = \hat{A}\hat{X}c$, d é a direção de deslocamento, r é o resíduo, e $s = (\hat{A}\hat{X}A^t)d$.

```
for i:=1 to n do  ( q_i := 0; y_i := 0 );
for j:=1 to m do
    begin
    if x_j>d_j-x_j then x̂_j:=x_j^2 else x̂_j:=(d_j - x_j)^2;
    q[t_j] := q[t_j] + x̂_j*c_j
    q[h_j] := q[h_j] + x̂_j*c_j
    end
r := q
d := r
while  inner(r,r) > ε  do
    begin
    s := mult(d)
    σ := inner(d,s)
    α := inner(r,d)/σ
    y := multsoma(y,α,d)
    r := multsoma(r,-α,s)
    β := -inner(r,s)/σ
    d = multsoma(r,β,d)
```

```
end
```

Onde

- inner(r,s)

```
∑ := 0
for i:=1 to n do ∑ := ∑ + r_i*s_i
inner := ∑
```

- mult(d)

```
s:=0
for j:=1 to m do
    begin
    s[t_j] := s[t_j] + x̂_j ( d[t_j] - d[h_j]
    s[h_j] := s[h_j] + x̂_j ( d[h_j] - d[t_j]
    end
mult := f
```

- multsoma(y, α, d)

```
s:=0
for i:=1 to n do s_i := y_i + α*d_i
multsoma := s
```

Os sistemas $(A\hat{Y}A^t)\Delta y = b - A\hat{Y}V^2u$ (seção 4.3.) e $(A\hat{X}^2A^t)h = -\hat{X}^2A^tB\rho$ (seção 4.2.) são resolvidos de modo similar.

5.7. Resolução de $(A\hat{X}A^t)y = A\hat{X}c$ pelo método gauss-seidel.

Neste caso é utilizado o vetor $\hat{x}[1..m]$ para armazenar os elementos da diagonal de $\hat{X}$. São também utilizados os vetores auxiliares $\hat{b}$, d, p, q onde $\hat{b} = A\hat{X}c$, p é o grau externo acumulado dos nós e q é o grau interno acumulado dos nós.

```
for i:=1 to n do  ( b̂_i := 0; y_i := 0 d_i := 0 );
for j:=1 to m do
      begin
      if x_j < (d_j - x_j) then x̂_j := x_j^2  else x̂_j := (d_j - x_j)^2;
      d[t_j] := d[t_j] + x̂_j
      d[h_j] := d[h_j] + x̂_j
      b̂[t_j] := b̂[t_j] + c_j * x̂_j
      b̂[h_j] := b̂[h_j] + c_j * x̂_j
      end
for i:=1 to n-1 do y_i := b̂_i / d_i
y_n := 0
Repeat
      for i:=1 to n-1 do
          begin
          som := b̂_i
          for j:=p[i-1] + 1 to p[i] do
              som := som + y[h[r_j]] * x̂[r_j]
          for j:=q[i-1] + 1 to q[i] do
              som := som + y[t[s_j]] * x̂[s_j]
          y[i] := som/d[i];
          end
Until     || y^k - y^(k-1) ||_∞ / || y^k ||_∞  <  ε
```

CAPITULO VI
RESULTADOS COMPUTACIONAIS

6.1. INTRODUÇÃO

Os métodos discutidos nos capítulos II, III, IV foram implementados em **TURBO PASCAL** 5.5 em um microcomputador **PC-AT386** Dx de 25 MHz com processador numérico. Os programas foram batizados como:

KILTER método "Out-of-kilter" (cap II)

PSIMP método primal simplex canalizado para redes (cap III)

DSIMP método dual simplex canalizado para redes (cap III)

PAFIM método primal afim escala canalizado (cap IV)+

DAFIM método dual afim escala canalizado (cap IV)+

PGRAD método primal afim escala canalizado para redes com gradiente conjugado (cap IV).

DGRAD método dual afim escala canalizado para redes com gradiente conjugado (cap IV).

PGAUSS método primal afim escala canalizado para redes com Gauss Seidel (cap IV).

Para avaliar o desempenho destes programas foram realizados testes comparativos com problemas de fluxo ótimo em redes gerados aleatoriamente de acordo com o seguinte esquema:

n, m, seed, ptrans: fornecidos para cada problema gerado. (n = número de nós, m = número de arcos, seed = semente de inicialização do gerador de números aleatórios, ptrans = probabilidade do nó i ser de transbordo; isto é, $b_i = 0$).

b_i: $b_i = 0$ com probabilidade ptrans; caso contrário é gerado independentemente com distribuição uniforme em (-10,10).

+ A implementação não está tirando proveito da esparsidade da matriz

$b_n = -\sum_{i \neq n} b_i$.

c_j = gerado independentemente com distribuição uniforme em (0,10).

d_j = gerado independentemente com distribuição uniforme em (0,10).

As redes foram gerados de acordo com o seguinte algoritmo:

```
read(n,m,ptrans,seed)
Begin
     s := 0;
     for i:=2 to n do
          if random(seed)<ptrans
             then  b_i := 0
             else
                  begin
                  b_i := 20*random(seed) - 10;
                  s := s + b_i
                  end
     b_1 := -s;
     for j:=1 to m do
          bogin
          repeat
               t_j := ⌊ n*random(seed) ⌋ + 1
               h_j := ⌊ n*random(seed) ⌋ + 1
          until ( t_j ≠ h_j ) and ( b_t_j ≤ 0 ) and ( b_h_j ≥ 0 )
          c_j := 10*random(seed)
          d_j := 10*random(seed)
          end
End
```

A seguir é dado o Espaço de Memória requerido por cada um dos métodos implementados.

Espaço de Memória requerido

	PSIMP	DSIMP	KILTER	PGRAD	DGRAD	PGAUSS	PAFIM	DAFIM
n-vetores	6	5	5	8	9	7	3	4
m-vetores	5	6	7	4	14	10	5	9
n×m matriz	0	0	0	0	0	0	1	1
n×n matriz	0	0	0	0	0	0	2	2

6.2. RESULTADOS

A seguir são apresentadas as tabelas e os gráficos elaborados com base nos resultados obtidos nos testes comparativos. Cada conjunto de testes é o resultado de 5 problemas factíveis gerados com os mesmos parâmetros mas com diferentes sementes. Para cada classe de problemas foram gerados diversos problemas de modo a garantir exatamente 5 problemas factíveis. Cada tabela a seguir apresenta o tempo de cpu médio (em centésimos de segundo) ou o número de iterações médio.

n=10 m=40								
	PSIMP	DSIMP	KILTER	PAFIM	DAFIM	PGRAD	DGRAD	PGAUSS
ITR	18.6	11.6	9.0	28.6	29.4	28.6	29.4	28.6
CPU	5.2	5.2	5.6	340.4	361.6	128.4	137.4	9868.8

CPU : Tempo [sec/100]

n	m	PSIMP	DSIMP	KILTER	PGRAD	DGRAD	PAFIM	DAFIM
30	70	6.4	13.2	23.2	395.4	624.0	4374.4	4276.4
30	80	7.6	11.0	25.0	510.6	819.6	4621.0	5305.6
30	90	11.0	16.8	26.8	508.6	819.8	6056.4	6270.2
30	100	11.0	18.8	34.0	573.2	952.0	7094.8	6922.6
30	110	11.0	19.8	34.2	549.4	995.2	8276.2	8078.6
30	120	16.4	27.4	37.6	641.2	1241.2	10008.2	9664.6
30	130	15.6	30.6	44.8	758.2	1272.0	10356.6	10684.0
30	140	21.0	29.6	46.0	916.4	1467.6	12282.0	------
30	150	22.0	34.2	51.6	784.6	1514.8	13369.8	------

No. Iterações

n	m	PSIMP	DSIMP	KILTER	PGRAD	DGRAD	PAFIM	DAFIM
30	70	65.0	43.8	30.0	16.2	25.6	26.2	24.8
30	80	64.6	41.8	27.6	19.6	31.4	23.8	27.2
30	90	71.6	49.6	28.2	18.2	29.2	28.0	28.8
30	100	71.8	49.8	29.6	19.2	31.8	29.4	28.8
30	110	75.8	56.4	28.6	17.2	31.2	31.6	30.6
30	120	86.2	54.0	28.0	18.6	35.6	33.8	33.0
30	130	86.0	58.2	31.2	21.0	34.6	32.4	33.8
30	140	103.6	52.6	29.0	24.0	37.8	35.8	----
30	150	104.4	56.0	30.0	19.6	37.2	36.4	----

CPU : Tempo [sec/100]						
n	m	PSIMP	DSIMP	KILTER	PGRAD	DGRAD
50	500	59.4	170.2	309.8	8552.8	11241.4
75	500	97.8	285.6	416.2	7740.0	16794.2
100	500	127.8	430.6	533.6	9865.6	17081.6
125	500	167.0	608.2	607.2	21625.8	19666.8

No. Iterações						
n	m	PSIMP	DSIMP	KILTER	PGRAD	DGRAD
50	500	244.0	134.4	51.6	48.0	62.8
75	500	398.4	219.6	80.4	27.2	59.0
100	500	475.4	375.2	104.4	24.4	42.2
125	500	549.2	446.0	131.6	42.6	36.4

CPU : Tempo [sec/100]						
n	**m**	**PSIMP**	**DSIMP**	**KILTER**	**PGRAD**	**DGRAD**
100	**400**	95.6	405.2	399.6	10828.2	12170.4
100	**450**	103.4	394.4	465.8	8827.8	14515.6
100	**500**	127.8	430.6	533.6	9865.6	17081.6
100	**550**	135.0	490.2	550.0	17147.8	19809.4
100	**600**	147.2	576.8	616.4	18170.0	21333.0

No. Iterações						
n	**m**	**PSIMP**	**DSIMP**	**KILTER**	**PGRAD**	**DGRAD**
100	**400**	383.0	360.6	2415.4	31.6	35.4
100	**450**	402.6	326.8	105.4	23.2	33.0
100	**500**	475.4	375.2	104.4	24.4	42.2
100	**550**	476.5	337.2	101.6	39.8	46.2
100	**600**	504.6	371.0	102.6	39.2	46.6

CPU : Tempo [sec/100]			
n m	PSIMP	DSIMP	KILTER
50 800	76.0	299.8	672.2
100 800	163.8	793.2	897.2
150 800	300.2	1367.4	1196.2
200 800	367.0	1536.4	1454.6

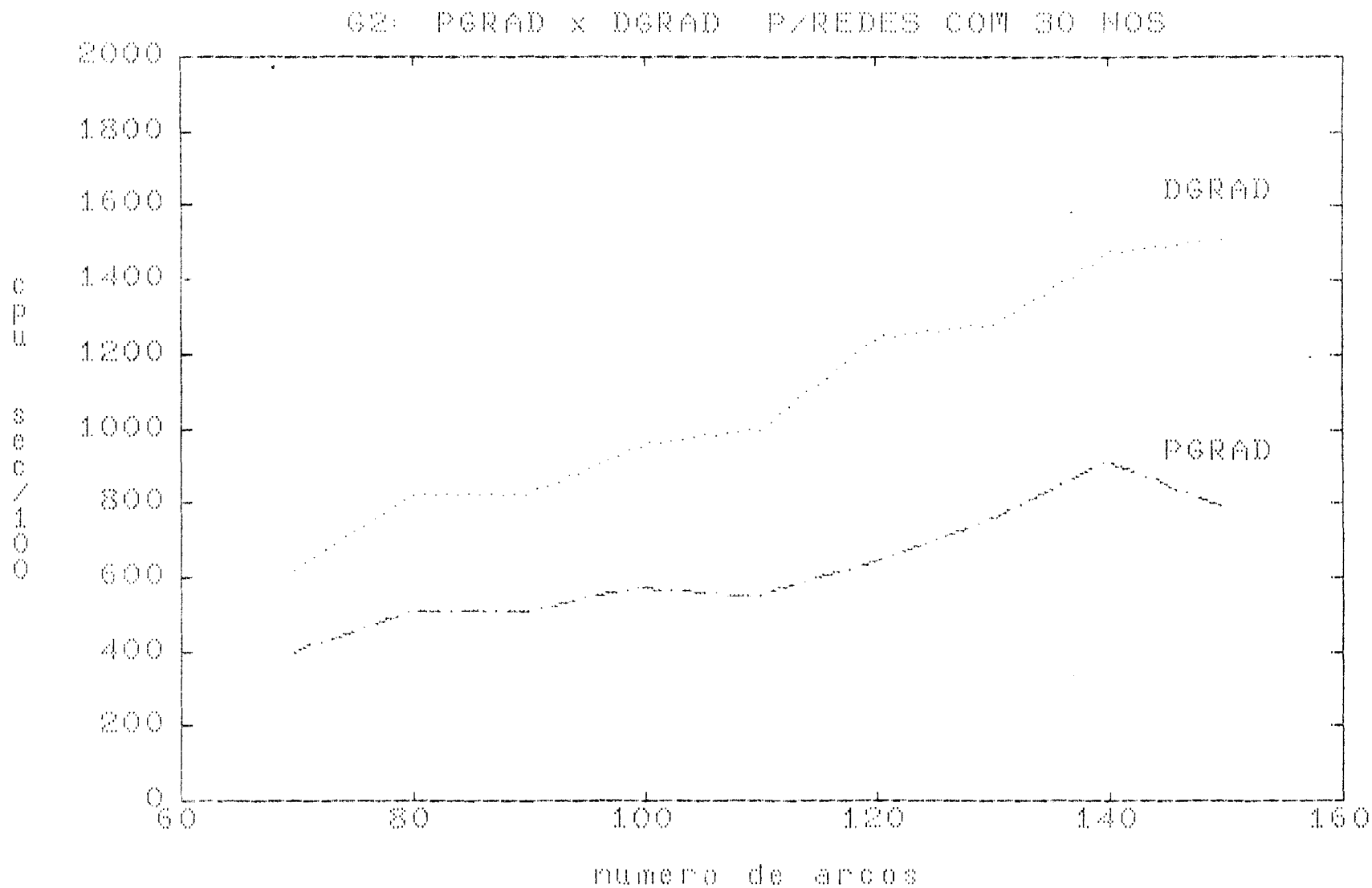
No. Iterações			
n m	PSIMP	DSIMP	KILTER
50 800	282.2	151.2	52.8
100 800	548.8	403.8	104.6
150 800	850.0	662.2	151.8
200 800	961.4	763.0	201.2

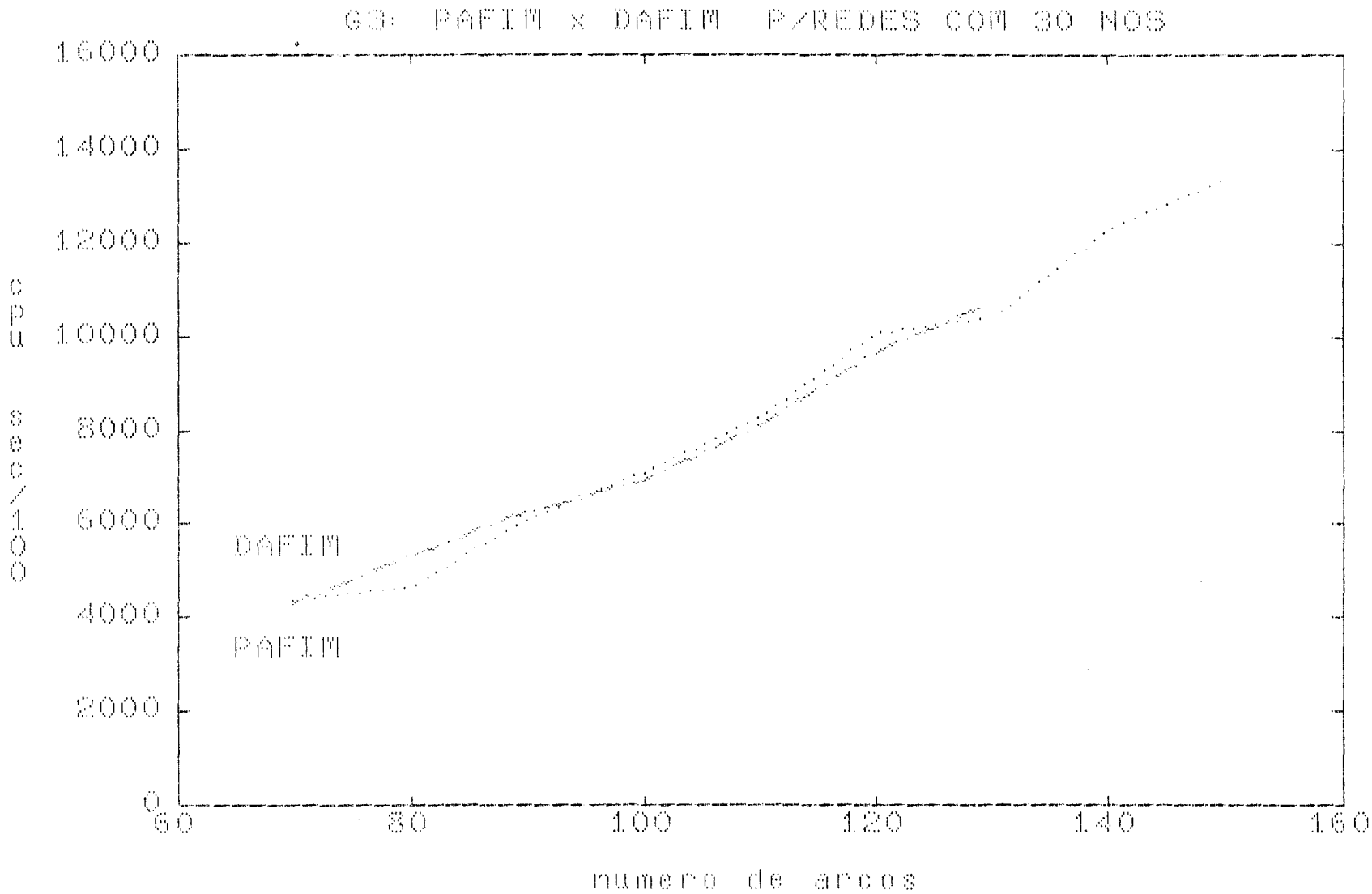
CPU : Tempo [sec/100]			
n m	PSIMP	DSIMP	KILTER
200 700	347.2	1348.8	1306.0
200 725	353.8	1430.4	1342.6
200 750	373.4	1562.0	1391.8
200 775	379.0	1911.6	1422.2
200 800	425.2	1769.6	1537.8

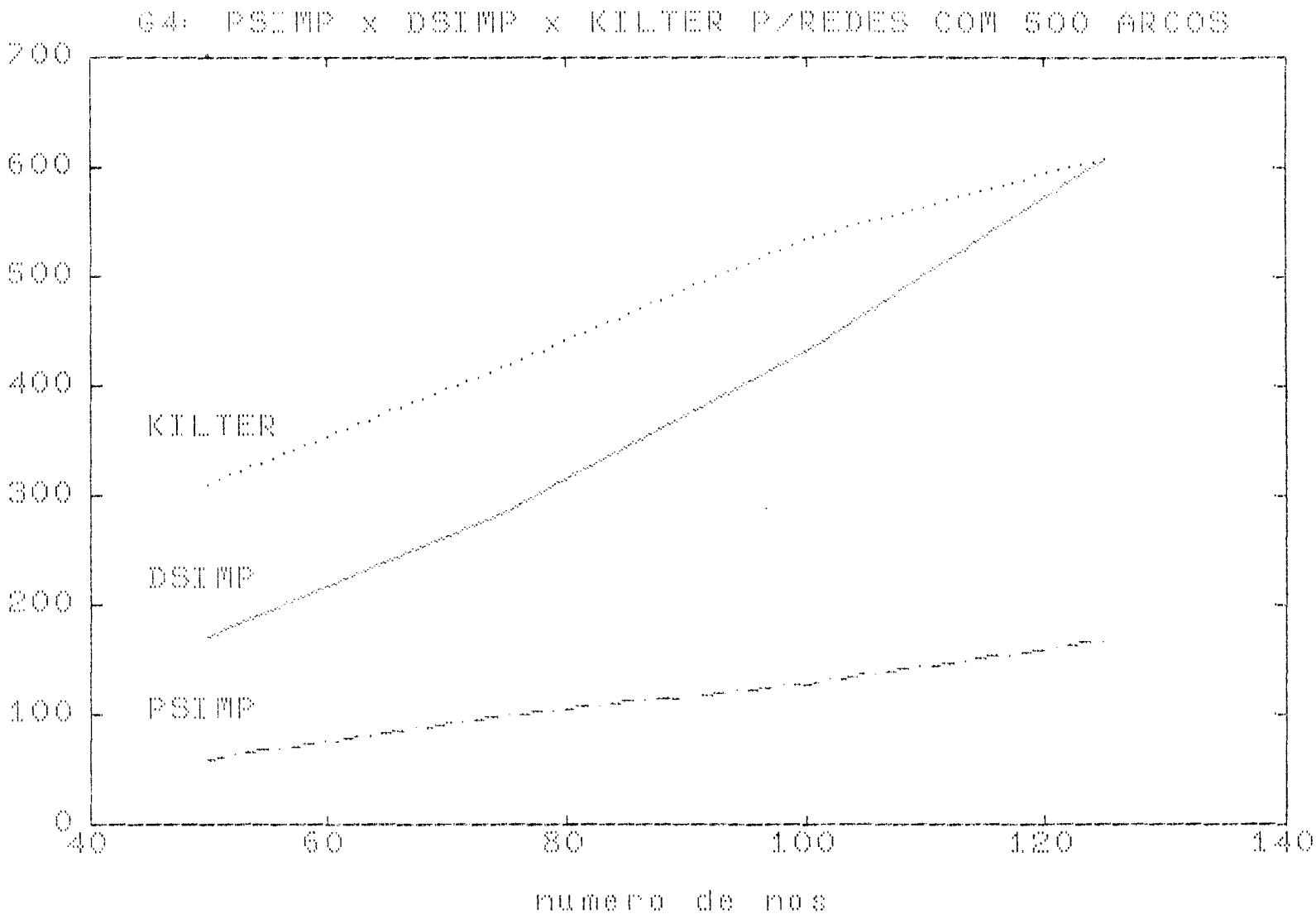
No. Iterações			
n m	PSIMP	DSIMP	KILTER
200 700	865.8	608.6	201.4
200 725	867.2	634.6	200.2
200 750	905.2	659.8	198.6
200 775	933.6	757.8	199.2
200 800	1028.6	704.2	200.4

A seguir são apresentados os gráficos relativos aos tempos de **cpu** em **sec/100** dados nas tabelas acima.

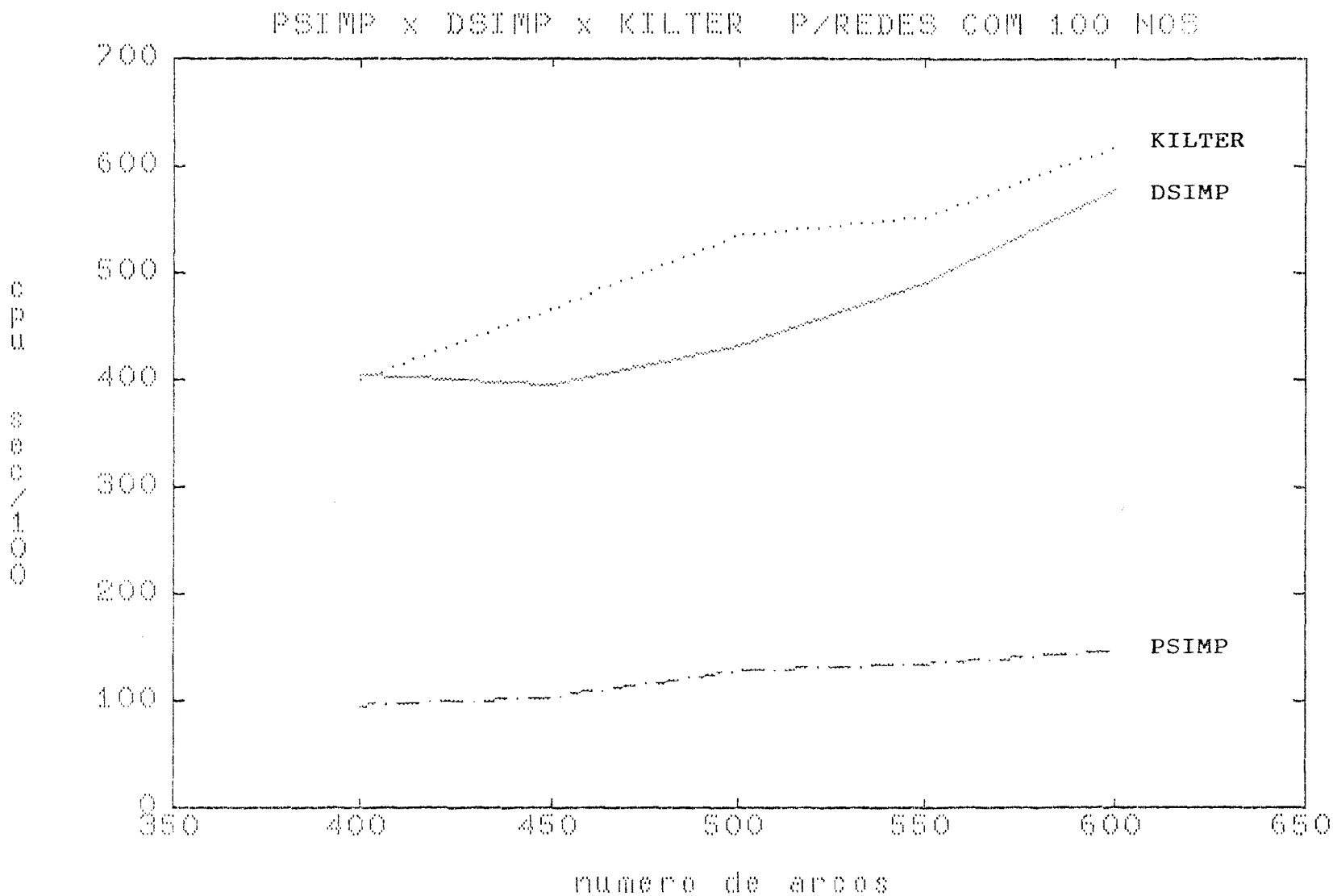
61: PSIMP x DSIMP x KILTER P/REDES COM 30 NOS
KILTER
DSIMP
PSIMP
60
50
40
30
20
10
60
80
100
120
140
160
numero de arcos
CPU segs/100

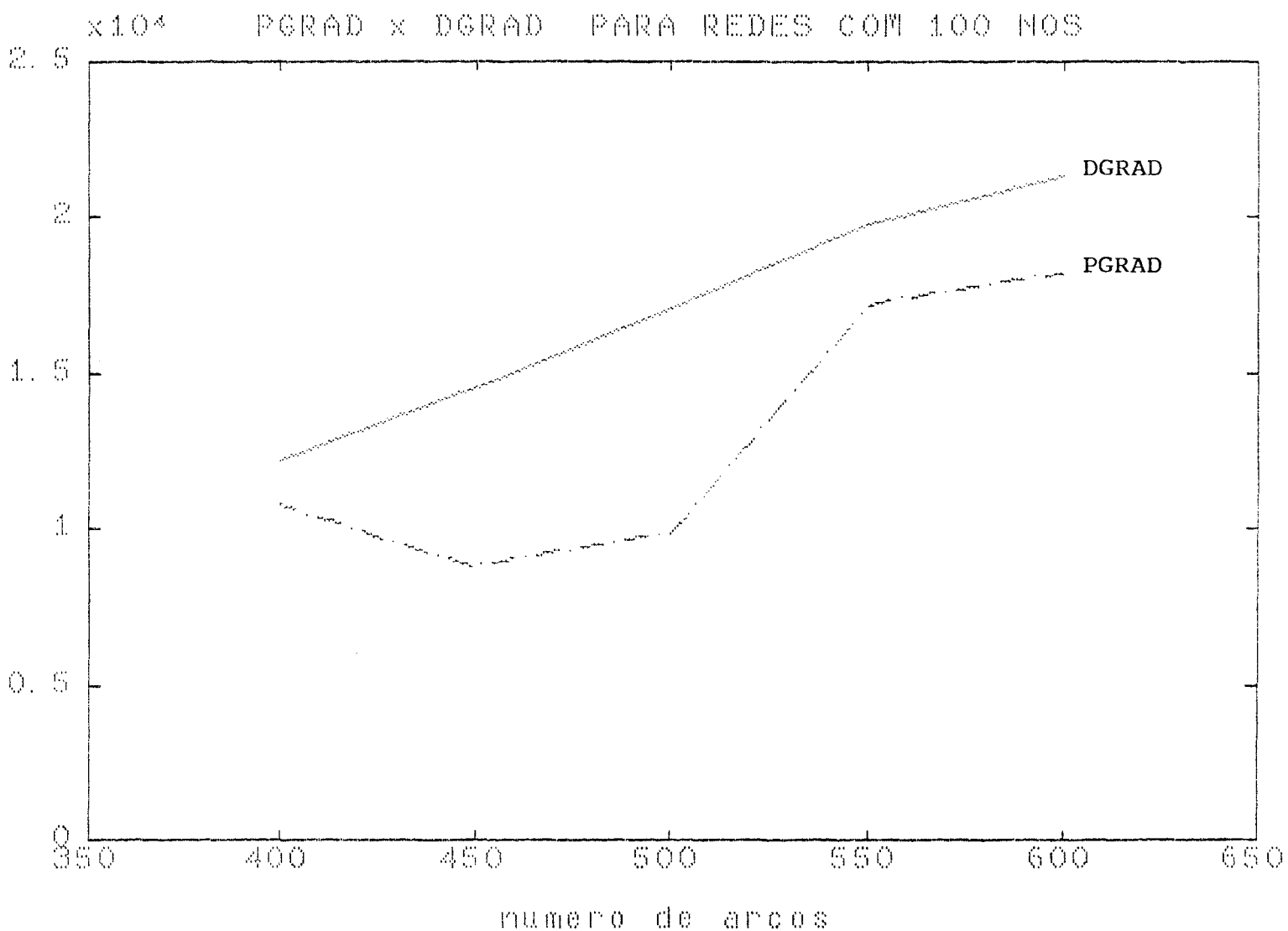

G2: PGRAD x DGRAD P/REDES COM 30 NOS
2000
1800
1600
1400
1200
1000
800
600
400
200
0
cpu sec/100
60
80
100
120
140
160
numero de arcos
DGRAD
PGRAD

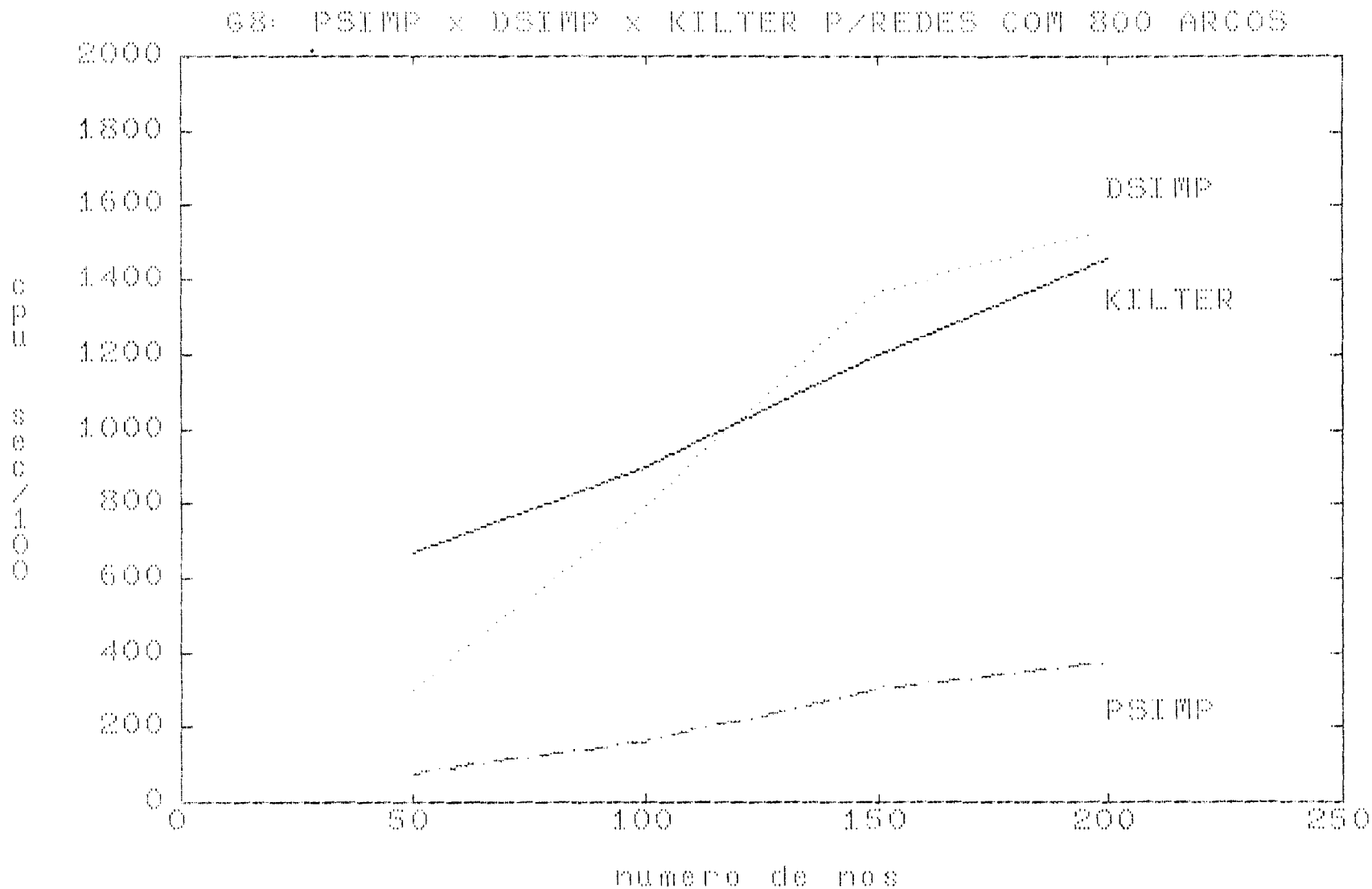

G3: PAFIM x DAFIM P/REDES COM 30 NOS
16000
14000
12000
10000
8000
6000
4000
2000
0
60
80
100
120
140
160
cpu/seg
numero de arcos
DAFIM
PAFIM

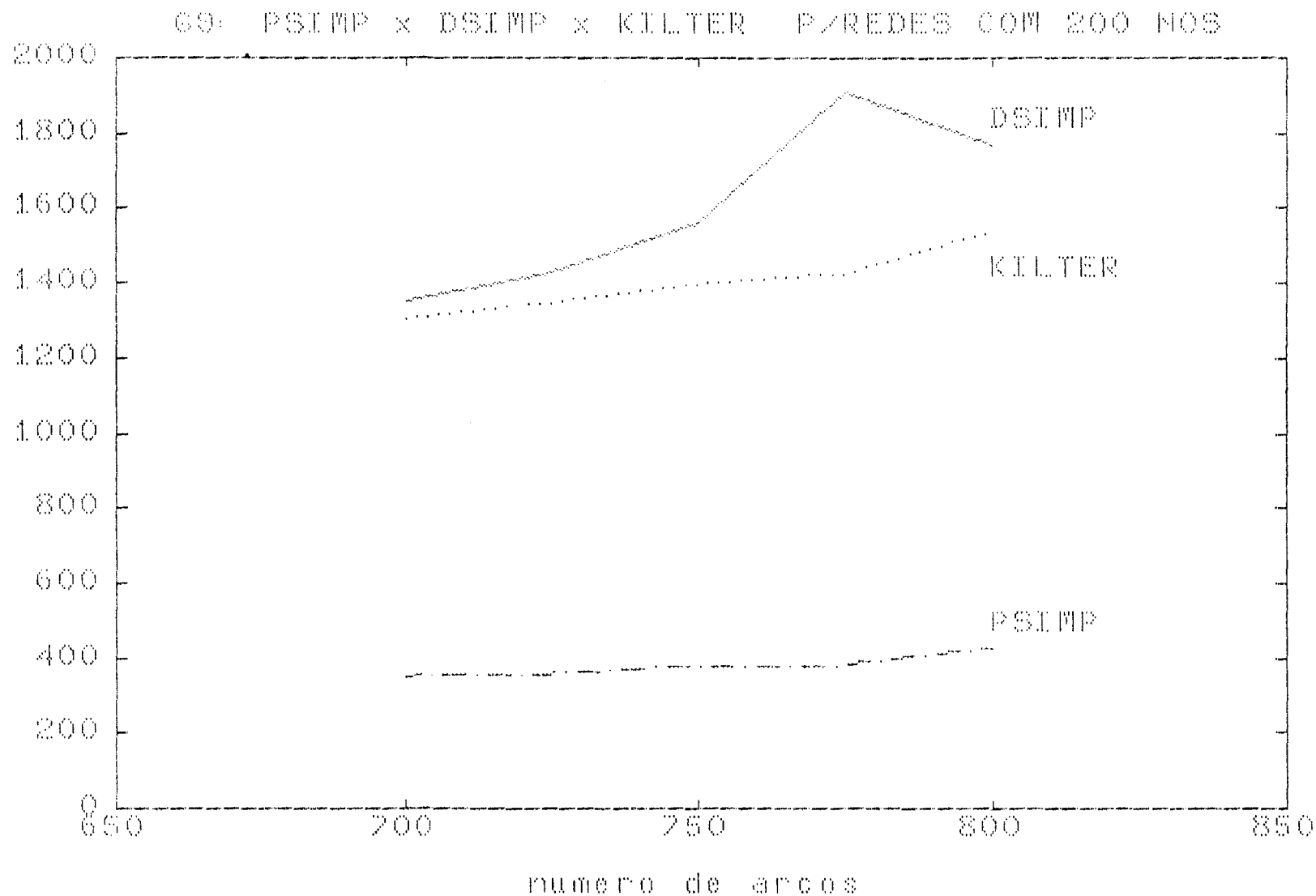

G4: PSIMP x DSIMP x KILTER P/REDES COM 500 ARCOS
700
600
500
400
300
200
100
0
cpu seg/100
KILTER
DSIMP
PSIMP
40
60
80
100
120
140
numero de nos

x10^4 GS: PGRAD x DGRAD P/REDES COM 500 ARCOS
2.5
2
1.5
1
0.5
40
60
80
100
120
140
DGRAD
PGRAD
número de nós


PSIMP x DSIMP x KILTER P/REDES COM 100 NOS
700
600
500
400
300
200
100
0
350
400
450
500
550
600
650
KILTER
DSIMP
PSIMP
cpu seg/100
numero de arcos


x10^4 PGRAD x DGRAD PARA REDES COM 100 NOS
2.5
2
1.5
1
0.5
0
350
400
450
500
550
600
650
DGRAD
PGRAD
numero de arcos


G8: PSIMP x DSIMP x KILTER P/REDES COM 800 ARCOS
2000
1800
1600
1400
1200
1000
800
600
400
200
0
cpu seg/100
0
50
100
150
200
250
numero de nos
DSIMP
KILTER
PSIMP


60: PSIMP x DSIMP x KILTER P/REDES COM 200 NOS
2000
1800
1600
1400
1200
1000
800
600
400
200
0
650
700
750
800
850
DSIMP
KILTER
PSIMP
cpu sec/100
numero de arcos

[100 600]	DGRAD	DGRAD MODIFICADO
ITR	46.2	46.2
CPU	21166.8	21950.0

DGRAD MODIFICADO

NUMERO DE ITERACOES DO GRADIENTE CONJUGADO (NIGC)					
itr	Prob1	Prob2	Prob3	Prob4	Prob5
1	34	45	35	43	34
2	41	48	42	46	41
3	40	44	42	45	40
4	40	43	40	43	40
5	42	45	44	41	35
6	69	60	54	60	44
7	98	71	94	69	72
8	100	98	100	82	88
9	100	99	100	97	100
10	100	100	100	100	100
...	...	...	...	...	...

total nigc	4664	4453	4251	3826	4094
no. itr	50	48	46	42	45
nigc/no.itr	93.28	92.77	92.41	91.09	90.97

número médio de nigc por iteração = 92.10 (problemas com 100 nós)

A seguir é dado o gráfico que representa o desempenho do Gradiente Conjugado Modificado relativo ao problema 1.

610 : DESEMPENHO DO GC
Interacao do DGRAD
0
2
4
6
8
10
0
20
40
60
80
100
120

Com base nos dados acima podem ser feitas as seguintes observações:

1. O programa **PGAUSS** apresenta o pior desempenho e com tempos de cpu exageradamente altos.
2. Os programas **PAFIM** e **DAFIM** não apresentaram um bom desempenho e não foi possível fazer a compilação destes programas para redes com 50 ou mais nós devido ao espaço de memória requerido. De modo geral apresentaram um comportamento semelhante.
3. Os programas **PGRAD** e **DGRAD** foram compilados para redes com 100 nós 600 arcos e apresentaram um desempenho satisfatório. **PGRAD** tende a apresentar o número de iterações e o tempo de cpu menores do que **DGRAD**.
4. Os programas **PSIMP, DSIMP, KILTER** apresentaram os melhores resultados. **PSIMP** tende a apresentar o tempo de **cpu** menor do que os outros dois.

Deve ser ressaltado que os programas **PSIMP, DSIMP, KILTER** trabalham apenas com variáveis inteiras (não utilizam o processador aritmético do microcomputador) e não executam operações de multiplicação e divisão no procedimento iterativo; multiplicações são efetuadas apenas para calcular o valor da função objetivo primal e dual, ao final do programa. Os demais programas **PAFIM, DAFIM, PGRAD, DGRAD, PGAUSS** apresentam desempenho e precisão mais fracas caso o microcomputador não disponha de processador aritmético.

Os programas **PAFIM, DAFIM, PGRAD, DGRAD, PGAUSS** necessitam de especificação de tolerâncias para definir o critério de parada; **PGAUSS** necessita uma segunda tolerância para definir o critério de parada de resolução do sistema de equações lineares. Foi utilizada a mesma tolerância para todos os programas, entretanto foi observado que os programas **PAFIM, PGRAD, PGAUSS** obtinham as soluções ótimas com maior

dificuldade.

O programa **DGRAD** foi alterado para permitir novas observações. O **DGRAD** original realiza a resolução do sistema de equações lineares sempre com n iterações do método gradiente conjugado. Para a nova versão foi introduzido uma segunda tolerância que permite interromper o método gradiente conjugado antes das n iterações; isto é feito quando a norma dos resíduos é muito pequena. Foi feito o levantamento do número de iterações do gradiente conjugado para cada iteração do método afim-escala para uma rede com n=100 nós e m=600 arcos. Foi observado que, inicialmente o número de iterações do gradiente conjugado é inferior a 100; posteriormente (acima de 10 iterações do método afim escala) o número de iterações do gradiente conjugado é mantido em n=100 indicando que o método afim escala tende a trabalhar com uma matriz (AXA^t) mal condicionada em quase todas as iterações.

6.3. CONCLUSÕES.

Em vista do exposto, podemos afirmar que os objetivos propostos no capítulo I foram alcançados. Foram estudados os principais métodos de resolução do problema de fluxo ótimo em redes, foram implementados alguns destes métodos e realizados experimentos computacionais para avaliar o desempenho destas implementações. É importante ressaltar que acreditamos que o método afim-escala sob a forma dos programas **PGRAD, DGRAD** venham a ter um comportamento comparativo muito melhor com o uso de redes muito maiores. O tamanho das redes utilizadas ficou limitado à memória de trabalho disponível no microcomputador.

Um importante estudo complementar que pode ser realizado é a análise da matriz $(A\hat{X}A^t)$ para o método Afim-escala/Gradiente cojugado. A matriz $(A\hat{X}A^t)$ torna-se mal condicionada à medida que as iterações do método afim-escala

vão sendo executadas. Já existem diversas técnicas de précondicionamento da matriz que podem ser analisadas (Vide Golub [22]). O objetivo é diminuir o número de iterações do gradiente conjugado a cada iteração do método afim-escala.

Um outro estudo igualmente importante se refere à análise do valor do parâmetro α dos métodos afins que impede que o novo ponto seja de fronteira. Foi utilizado $\alpha = 0.95$ mas a literatura científica também apresenta valores como $\alpha = 0.90$ ou $\alpha = 0.99$, os quais não foram utilizados neste trabalho.

Finalmente pode ser efetuada uma análise com relação ao valor da tolerância utilizada para o teste de parada dos algoritmos afins. Durante a etapa de desenvolvimento, verificou-se que os métodos duais afins permitiam uma tolerância menor do que os primais afins. Entretanto, o uso de valores muito pequenos para qualquer um dos programas fazia com que a solução não convergisse para uma solução ótima ou com que o valor da função objetivo se torne não decrescente (para os métodos primais) ou não crescente (para os métodos duais). Optou-se por utilizar um único valor para a tolerância: 10^{-6}.

BIBLIOGRAFIA

[1] I. Adler, M. G. C. Resende, G. Veiga and N. Karmarkar. *An Implementation of Karmarkar's Algorithm for Linear Programming*". Mathematical Programming, 44:297-335, 1989.

[2] D. Avis and V. Chvátal, "*Notes on Bland's Pivoting Rule*", Mathematical Programming Study 8, North-Holland, Amsterdam, 24-34, July 1978.

[3] E. R. Barnes. "*A Variation on Karmarkar's Algorithm for Solving Linear Programming Problems*", Mathematical Programming 36:174-182, 1986.

[4] K. Bayer and J. C. Lagarias. "*The Non-linear Geometry of Linear Programming, I. Affine and Projective Scaling Trajectories, II. Legendre Transform Coordeinates, III. Central Trajectories*". Preprints, AT&T Bell Laboratories, Murray Hill, NJ, 1986.

[5] M. S. Bazaraa and J. J. Jarvis, "*Linear Programming and Network Flows*", John Wiley & Sons, 1977.

[6] R G. Bland, D. Goldfarb and M. J. Todd, "*The Ellipsoid Method: A Survey*". Operations Research, 29, 6:1039-1091, 1981.

[7] J. R. Bunch and D. J. Rose. "*Sparse Matrix Computations*". Academic Press Inc. 1976.

[8] M.F. H. Carvalho, "*Modelos de Fluxo em Redes Aplicados a Sistemas de Energia Elétrica*". Tese de Doutorado, FEE UNICAMP 1986.

[9] V. Chandru, B. S. Kochar, "*A Class of Algorithms for Linear Programming*". Research Memorandum No. 85-14, Purdue University 1985.

[10] G. B. Dantzig and D. R. Fulkerson, "*Computation of Maximal Flows in Networks*". Naval Research Logistics Quarterly, 2:277-283, 1955.

[11] G. B. Dantzig, "*Linear Programming and Extensions*", Princeton University Press, 1963.

[12] G. B. Dantzig and D. R. Fulkerson, "*Computation of Maximal Flows in Networks*". Naval Research Logistics Quarterly, 2:277-283, 1955.

[13] I. I. Dikin. "*Iterative Solution of Problems of Linear and Quadratic Programming*". Soviet Mathematics Doklady, 8:674-675, 1967.

[14] I. I. Dikin. "*On the Speed of an Iterative Process*". Upravlyaemye Sistemi, 12:54-60 1974.

[15] I. S. Duff, A. M. Erisman and J. K. Reid, "*Direct Method for Sparse Matrices*", Clarendon Press 1986.

[16] M. Fiedler. "*Special Matrices and Their Aplications in Numerical Mathematics*". Martinus Nijhoff Publishers. 1986.

[17] R. Fletcher, "*Practical Methods of Optimization, Vol. 2, Constrained Optimization*", John Wiley & Sons, 1986.

[18] D. R. Fulkerson, "*An Out-of-Kilter Method for*

Minimal-Cost Flow Problems". J. Soc. Indust. Appl. Math., 9, 1:18-27, 1961.

[19] D. Gay. "*A Variant of Karmarkar's Linear Programming Algorithm for Problems in Standard Form*". Mathematical Programming, 37:81-89, 1987.

[20] D. Goldfarb & S. Mehrotra. "*Relaxed Variants of Karmarkar's Algorithm for Linear Programs with Unknown Optimal Objetive Value*". Mathematical Programming 40 (1988), 183-195.

[21] D. Goldfarb and W. Y. Sit, "*A Practicable Steepest-Edge Simplex Algorithm*", Mathematical Programming, 12 (1977), 361-371.

[22] G. H. Golub, C. F. Van Load. "*Matrix Computations*". The Jhons Hopkins University Press, 1983.

[23] C. Gonzaga. "*An Algorithm for Solving Linear Programming Problems in $O(n^3L)$ operations*".Memoramdum UCB/ERL M87/10, Electronics Research Laboratory, University of California, Berkeley, CA, March 1987.

[24] C. Gonzaga. "*Algoritmos de Pontos Interiores para Programação Linear*", 17o. Colóquio Brasileiro de Matemática, IMPA - RJ, Julho, 1989.

[25] C. C. Gonzaga. "*Conical Projection Algorithms for Linear Programming*". Mathematical Programming, 43:151-153, 1989.

[26] P. M. J. Harris. "*Pivot Selection Methods of the DEVEX LP Code*", Mathematical Programming, 5, 1-28, 1973.

[27] E. Isaacson H. B. Keller. "*Analysis of Numerical Methods*" . John Wiley &. Sons, INC. 1966.

[28] I. Jacques and C. Judd. "*Numerical Analysis*". London New York Chapman and Hall. 1980.

[29] N. Karmarkar, "*A New Polynomial-Time Algorithm for Linear Programming*", Combinatorica 4:373-395, 1984.

[30] L. G. Khachiyan, "*A Polynomial Algorithm in Linear Programming*", Soviet Math-Doklady 20:191-194, 1979.

[31] J. L. Kennington and R. V. Helgason, "*Algorithms for Network Programming*", John Wiley & Sons, 1980.

[32] V. Klee and G. J. Minty, "*How Good is the Simplex Algorithm ?*", In O. Shisha, ed. Inequalities-III 1972, 159-175, Academic Press, New York.

[33] D. E. Knuth. "*The Art of Computer Programming vol. 1 Fundumental Algorithms*". Addison Wesley Publishing Company 1973.

[34] M. Kojima, S. Mizuno, and A. Yoshise. "*A Primal-Dual Interior Point Method for Linear Programming*". Research Report B-188, Dept. of Information Sciences, Tokyo Institute de Technology, Tokyo, Japan, 1987.

[35] D. G. Luenberger, "*Introduction to Linear and Nonlinear Programming*", Addison-Wesley Publishing Company, Inc., 1973.

[36] N. Megiddo. "*Pathways to the Optimal Set in Linear*

Programming". Research Report RJ 5295, IBM Almaden Research Center, S. Jose, California, 1986.

[37] R. C. Monteiro and I. Adler. "*An $O(n^3L)$ Interior Point Algorithm for Convex Quadratic Programming*". Manuscript, Dept of Industrial Engineering and Operations Research, University of California, Berkeley, CA, 1987.

[38] R. C. Monteiro and I. Adler. "*An $O(n^3L)$ Primal Dual Interior Point Algorithm for Linear Programming*". Manuscript, Dept of Industrial Engineering an Operation Research, University of California, Berkeley, CA, 1987.

[39] K. G. Murty. "*Linear Programming*", John Wiley & Sons, 1983.

[40] C. H. Papadimitriou and K. Steiglitz, "*Combinatorial Optimization: Algorithms and Complexity*". Prentice-Hall 1982.

[41] A. Ralston. "*Introduccion al Analisis Numérico*". Limusa, Wiley S.A. 1970.

[42] J. Renegar. "*A Polynomial-time Algorithm Based on Newton's for Linear Programming*". Report MSRI 07 118-86, Mathematical Sciences Research Institute, Berkeley, California, June 1986.

[43] M. G. C. Resende, G. Veiga. "*A Dual Affine Scaling Algoritm for Minimum Cost Network Flow*". Report Internal version 1.0-November 19, 1990.

[44] D. J. Rose and R. A. Willoughby. "*Sparse matrices and*

their Applications". Penum Press-New York-London. 1972.

[45] P. M. Vaidya. "*An Algorithm for Linear Programming which Requires* $O(((m+n)n^2 + (m+n)^{1.5}n)L)$ *Arithmetic Operations*". Preprint, AT&T Bell Laboratories, Murray Hill, NJ, 1987.

[46] R. J. Vanderbei. "*Affine-Scaling for Linear Programs with Free Variables*". Mathematical Programming, 43:31-44, 1989.

[47] R. J. Vanderbei, M. J. Meketon, and B. A. Freedman". *A Modification of Karmarkar's Linear Programming Algorithm*". Algorithmica, 1:395-407, 1986.

[48] Y. Ye. "*Karmarkar's Algorithm and the Ellipsoid Method*". Operations Research Letters, 4:177-182, 1987.

[49] G. Zoutendijk. "*Mathematical Programming Methods*". North-Holland Publishing Company Amsterdam. 1976.