UNIVERSIDADE ESTADUAL DE CAMPINAS

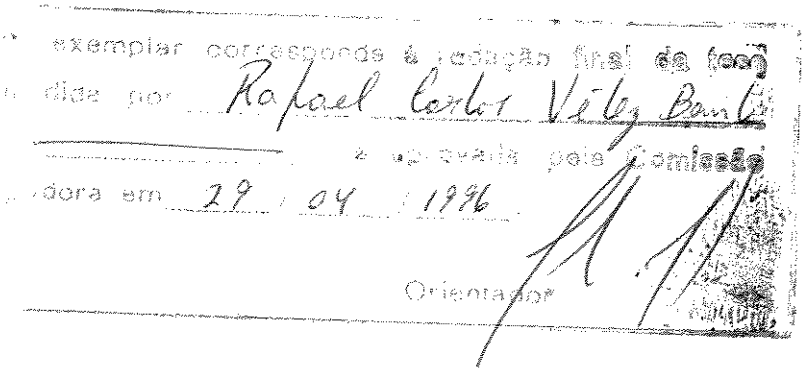
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE SISTEMAS

Uma Contribuição à Solução de Problemas de Fluxo de Custo Mínimo Através de Métodos de Pontos Interiores

Por: Rafael Carlos Vélez Benito

Orientador: Prof. Christiano Lyra Filho



Tese defendida e aprovada na Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do Título de Doutor em Engenharia Elétrica.

Setembro 1996

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

V543c

Vélez Benito, Rafael Carlos
Uma contribuição à solução de problemas de fluxo de custo mínimo através de métodos de pontos interiores / Rafael Carlos Vélez Benito .--Campinas, SP: [s.n.], 1996.

Orientador: Christiano Lyra Filho.
Tese (doutorado) - Universidade Estadual de Campinas Faculdade de Engenharia Elétrica e de Computação.

1. Pesquisa operacional. 2. Otimização combinatória. 3. Pesquisa matemática. I. Lyra Filho, Christiano. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

A minha esposa Nellita...

A minhas filhas Ghandy e Sandy...

Aos meus pais Rafael e Fortunata...

Agradecimentos

Ao *Prof. Christiano Lyra Filho*, pela orientação e dedicação constantes durante todo o desenvolvimento deste trabalho.

Ao *Prof. Clovis Perin Filho*, pelas suas sugestões sempre oportunas.

A *minha linda família* pelo estímulo e compreensão constantes.

Aos *professores e amigos do Densis*, pelo apoio.

Ao *povo brasileiro* que mantém universidades públicas e órgãos de apoio à pesquisa.

Ao *Conselho Nacional de Desenvolvimento Científico e Tecnológico* pelo apoio financeiro.

Ao *Centro Nacional de Processamento de Alto Desempenho em São Paulo*, pelo apoio à pesquisa.

À *Fundação de Amparo à Pesquisa do Estado de São Paulo*, pelo apoio à pesquisa.

Resumo

O presente trabalho faz um estudo cuidadoso dos métodos de pontos interiores para obter implementações eficientes na solução de problemas de fluxo de custo mínimo.

Tendo em vista que a maior parte do esforço computacional dos algoritmos baseados nos métodos de pontos interiores é dedicado à solução de sistemas do tipo $AD^*A^Ty = \hat{b}$, é feita uma análise deste sistema, explorando-se as características da estrutura de redes. Implementa-se especializações dos métodos diretos e dos métodos iterativos.

Os métodos diretos são especializações da decomposição de Choleski. Heurísticas do tipo *grau mínimo* e *preenchimento local mínimo* são usadas para reordenação das linhas e colunas, procurando conservar a esparsidade da matriz AD^*A^T. Para a família dos problemas de transportes e atribuição, desenvolve-se um método de ordenação ótima.

Os métodos iterativos são do tipo gradiente conjugado pré-condicionado. A estrutura de rede permite agilizar o cálculo das direções, reduzir requisitos de memória e construir pré-condicionadores *bem informados*. Um pré-condicionador do tipo diagonal é usado nos primeiros passos dos métodos de pontos interiores. Quando a solução se aproxima da otimalidade, constrói-se um outro pré-condicionador, baseado em estimações sobre a base ótima.

Desenvolve-se implementações especializadas à problemas de fluxo de custo mínimo dos métodos primal afim, dual afim, primal dual e preditor-corretor.

Interpretações baseadas no método de Newton para solução de problemas não lineares levaram a inovações nas implementações dos métodos afins.

Estuda-se o problema de *falta de volume* (ou falta de pontos interiores), aspecto frequente em problemas de fluxo de custo mínimo. Avalia-se suas conseqüências na utilização de métodos de pontos interiores.

A partir dos experimentos computacionais com os códigos desenvolvidos, procura-se fazer uma sistematização dos problemas em classes, com indicação das melhores alterna-

tivas de solução para cada classe.

Finalmente, faz-se extensões das idéias desenvolvidas para a resolução de problemas de fluxos generalizados, através de métodos de pontos interiores.

Abstract

This work is a careful study of the interior point methods looking for efficient implementations for network flow linear programs. Computational experiments are developed with the primal affine, dual affine, primal dual and predictor-corrector methods looking for the best alternatives for different classes of problems. We discuss Newton's method interpretations of these methods.

Since most of the computational time of the interior point methods is spent solving systems of the type $(AD^*A^T)y = \hat{b}$, for different diagonal matrices D^* and the same incidence matrix A we take advantage of the structure of such systems for networks.

We use the sparse Cholesky factorization with two heuristics: the minimum degree and local minimum fill-in. For the family transportation and assignment problems, it is developed an optimal ordering method.

We also use the conjugate gradient method with diagonal and maximum spanning tree preconditioners.

We give particular attention to the degenerescency phenomena mainly to the lack of primal and/or dual interior feasible points. We study their consequences to interior point methods.

Finally proposes extensions of the main ideas to the generalized network problems.

Conteúdo

Capítulo 1

Introdução e Objetivos

1.1 Introdução

No presente trabalho, estamos interessados nos métodos de solução de problemas de Programação Linear especializados ao fluxo por uma rede a custo mínimo. Estes problemas têm um grande número de aplicações em diversas áreas importantes, como por exemplo, produção-distribuição, logística, tráfego urbano, sistemas de comunicações, fluxos em redes de distribuição de água, localização de recursos e fluxos de energia em redes elétricas.

Desde o desenvolvimento do método simplex por George B. Dantzig em 1947, que os modelos lineares de fluxos em redes têm sido estudados. O problema de fluxo de custo mínimo é um programa linear especial, cuja estrutura simplifica os métodos de resolução, tornando-os mais eficientes computacionalmente tanto em tempo de execução quanto em memória requerida. Além disto, a geometria de uma rede pode ser facilmente representada através de desenhos a duas dimensões.

Este trabalho tem por objetivo realizar um estudo dos métodos de pontos interiores e propor alternativas de implementações para resolver o Problema de Fluxo de Custo Mínimo (PFCM), comparando seu desempenho com o método simplex. Procura também criar algumas referências que permitam indicar os algoritmos mais vantajosos (pontos interiores ou simplex) para solução de um dado problema.

1.2 Problema de Fluxo de Custo Mínimo (PFCM)

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

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

$$\begin{array}{ll} Minimizar & c^T x \\ sujeito\ a & Ax = b \\ & l \leq x \leq d \end{array} \tag{1.1}$$

onde A é uma matriz de incidência com m linhas (aos quais estão associados os nós do grafo) e n colunas (às quais estão associados os arcos do grafo). As colunas de A têm dois elementos não nulos, $+1$ na linha correspondente ao nó origem do arco e -1 na linha correspondente ao nó destino do arco.

Exemplo 1: Considere uma rede com 5 nós e 8 arcos (vide Figura 1.1). Os nós 1, 2 e 3 são nós produtores ($b_1 = 2$, $b_2 = 5$ e $b_3 = 1$), ao passo que os nós 4 e 5 são nós consumidores ($b_4 = b_5 = -4$). Além disto, a Figura 1.1 também mostra os custos associados aos arcos.

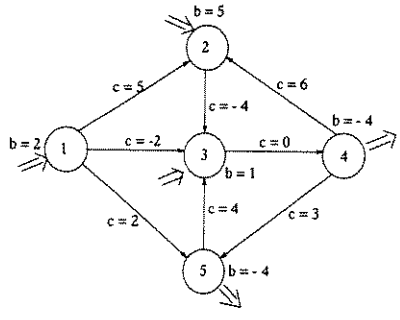


Figura 1.1: Exemplo 1 (5 nós e 8 arcos)

Escrevendo a formulação matemática, temos:

$$
\begin{array}{lcl}
Minimizar & \begin{pmatrix} 5 & -2 & 2 & -4 & 0 & 6 & 3 & 4 \end{pmatrix} & x \\
 & & \\
Sujeito\ a & \begin{pmatrix} 1 & 1 & 1 & & & & & \\ -1 & & & 1 & & -1 & & \\ & -1 & & -1 & 1 & & & -1 \\ & & & & -1 & 1 & 1 & \\ & & -1 & & & & -1 & 1 \end{pmatrix} & x = \begin{pmatrix} 2 \\ 5 \\ 1 \\ -4 \\ -4 \end{pmatrix} \\
 & & \\
 & & x \geq 0
\end{array}
$$

Observe que existe uma linha da matriz A para cada nó e que o arco $(1,2)$ apresenta coeficientes não nulos $+1$ na linha 1 e -1 na linha 2 e apresenta coeficiente de custo 5 na função objetivo; além disto, o nó 1 na primeira linha de A produz 2 unidades de fluxo.

As classes de problemas de otimização linear em redes descritas a seguir são casos particulares de (1.1).

- *Transporte*: cada nó ou é produtor ou consumidor; além disto, cada arco é direcionado de um nó produtor para um nó consumidor, formando uma rede bipartida.
- *Designação*: problema de transporte no qual $b_i = \pm 1\ \forall\ i \in \mathcal{N}$, formando uma rede com igual número de nós produtores e consumidores.
- *Circulação*: $b = 0$.
- *Fluxo Máximo* do nó s ao nó t em G: $b = 0, d_{ts} = \infty, c_{ts} = -1, c_{ij} = 0\ \forall\ (i,j) \neq (t,s)$.
- *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 particularidades das estruturas e cujo estudo e desenvolvimento podem ser encontrados na literatura clássica de programação linear [25, 64, 82, 58].

Todo problema de fluxo de custo mínimo (1.1) pode ser facilmente transformado em um problema de fluxo de custo mínimo com $l = 0$. Assim, sem perda de generalidade, neste trabalho vamos considerar o problema formulado com $l = 0$.

É fácil verificar que o posto de A é no máximo $m-1$; basta verificar que a soma das linhas de A é o vetor nulo. Desta forma, $\sum b_i = 0$ é uma condição necessária para que o problema (1.1) possa apresentar uma solução x satisfazendo $Ax = b$.

1.3 O Método Simplex

Nesta seção é apresentado o método simplex (primal), para resolver um problema de programação linear. Considere o programa linear de obter um n–vetor $x = (x_j)$, solução ótima de,

$$\begin{array}{lrcl} Minimizar & c^T x & & \\ sujeito\ a & Ax & = & b \\ & x & \geq & 0 \end{array} \qquad (1.2)$$

onde A é uma $m \times n$–matriz de posto completo e $m < n$; $b = (b_i)$ é um m–vetor; $c = (c_j)$ é um n–vetor.

Suponha que das n colunas da matriz A selecionamos um subconjunto de m colunas linearmente independentes, e denote por $B(m \times m)$ a matriz formada por essas colunas. A matriz B é não singular ($posto(A) = m$). Suponha sem perda de generalidade que B corresponde às primeiras colunas de A. Neste caso, a matriz A e os vetores x, c são particionados em:

$$A = [B|N] \quad x = \begin{pmatrix} x_B \\ x_N \end{pmatrix} \quad c = \begin{pmatrix} c_B \\ c_N \end{pmatrix}$$

Diz-se que o vetor x é uma *solução básica* de (1.2), se $x_N = 0$, $x_B = B^{-1}b$, o que implica $Ax = b$. Se $x_B \geq 0$, então diz-se que o vetor x é uma *solução básica factível* de (1.2).

Um vetor x que satisfaz $Ax = b$ é denominado *solução.* Uma solução que satisfaz $x \geq 0$ é denominada *solução factível.*

Uma solução factível x que atinge o valor mínimo da função objetivo ($c^T x$), é dita uma *solução factível ótima.*

O *Teorema Fundamental da Programação Linear* [64], ressalta a importância das soluções básicas factíveis na solução do problema (1.2).

- Se (1.2) tem uma solução factível, então ele tem uma solução básica factível.
- Se (1.2) tem uma solução factível ótima, então ele tem uma solução básica factível ótima.

Desta forma, para encontrar uma solução ótima de (1.2) pode-se limitar a procura entre as soluções básicas factíveis, que por serem em número limitado garantirão a sua obtenção em um número finito de avaliações.

O método simplex evolui iterativamente através de soluções básicas factíveis, melhorando a função objetivo. A cada iteração, verifica-se a condição de otimalidade,

$$z = \begin{pmatrix} z_B \\ z_N \end{pmatrix} = \begin{pmatrix} 0 \\ c_N - N^T y \end{pmatrix} \geq 0$$

onde $z = c - A^T y$ é o vetor de custos relativos. Se ela é violada para alguma variável, isto é,

$$\exists A_s \in N \quad tal \ que \quad z_s = c_s - A_s^T y < 0$$

a solução corrente não é ótima. Isto significa que o aumento da variável x_s melhora o valor da função objetivo, pois z_s é a componente do vetor de custos na direção de deslocamento associado a A_s.

O crescimento de x_s deve ser compensado pelas variáveis básicas, de modo que as restrições funcionais continuem satisfazendo as restrições de igualdade. Assim, a direção de movimento da solução Δx é expressa por,

$$\Delta x_k = 0 \quad k \in N, \quad k \neq s, \quad \Delta x_s = 1 \quad \Delta x_B = -\hat{A}_s = -B^{-1} A_s$$

de tal forma que,

$$B^{-1}(A\Delta x) = B^{-1}(B\Delta x_B + N\Delta x_N) = \Delta x_B + \hat{A}_s \Delta x_s = -\hat{A}_s + \hat{A}_s = 0$$

Admitindo que o problema (1.2) é limitado, o crescimento de x_s acarreta o decréscimo de pelo menos uma das variáveis básicas, havendo um bloqueio quando ela se

anula. Este procedimento é denominado *teste de razão*:

$$r = argmin\{\hat{b}_i/\hat{A}_{is} : \hat{A}_{is} > 0\}$$

Isto origina uma troca de base, com A_s substituindo A_r na base B. A nova solução é determinada por,

$$x + \alpha\Delta x$$

onde α é o tamanho de passo e é calculado por,

$$\alpha = \frac{\hat{b}_r}{\hat{A}_{rs}}$$

Assim, garante-se que $Ax = b$, $x \geq 0$.

Caso o problema (1.2) seja ilimitado, nenhuma variável básica bloqueia o crescimento de x_s. Neste caso, $\hat{A}_{is} \leq 0\ \forall i$.

O método simplex itera até que a condição de otimalidade ou que a condição de solução ilimitada seja satisfeita.

A estrutura lógica do método simplex é resumida a seguir.

Seja B uma base factível	$[B^{-1}b \geq 0]$
`Repita até parar`	
$x_N \leftarrow 0$	[Variáveis não básicas]
$x_B \leftarrow \hat{b} = B^{-1}b$	[Variáveis básicas]
$y^T \leftarrow c_B^T B^{-1}$	[Variáveis duais]
$z \leftarrow c - A^T y$	[Custos relativos]
`Se` $z \geq 0$ `então pare`	[Solução ótima]
`Seja` $z_s < 0$	[Entra A_s]
$\hat{A}_s \leftarrow B^{-1}A_s$	[Coluna s atualizada]
$\alpha \leftarrow Min[\hat{b}_i/\hat{A}_{is} : \hat{A}_{is} > 0]$	[Teste de razão]
`Se` $\alpha = \infty$ `então pare`	[Solução ótima ilimitada]
`Atualize` B , B^{-1}	[Sai A_r]

Observação: Na prática B^{-1} não é explicitamente calculada, sendo mais eficiente manter a sua decomposição LU e atualizar $\hat{b}$, $\hat{A}_s$ sempre que necessário.

Eventualmente não existe nenhuma base factível inicial para o problema (1.2); neste caso trata-se de um problema infactível. É comum aplicar o método fase 1 do simplex para achar uma solução inicial básica factível resolvendo o problema artificial,

$$Minimizar \ \{ \ \sum w_i : Ax + w = b \ \ x, w \geq 0 \ \}$$

a partir da solução inicial $x = 0$, $w = b$ com base inicial $B = I$.

O maior esforço computacional do método simplex a cada iteração, está no cálculo das variáveis duais ($y^T = c_B^T B^{-1}$), na obtenção da coluna atualizada ($\hat{A}_s = B^{-1} A_s$), no teste de razão ($\alpha = Min[\hat{b}_i / \hat{A}_{is} : \hat{A}_{is} > 0]$), e principalmente, na atualização da inversa da matriz básica (B^{-1}). São nestas etapas que os algoritmos especializados para grafo economizam esforços, aproveitando as particularidades das estruturas da base B e da coluna A_s.

1.3.1 Simplex em Rede

A matriz A associada a um problema de fluxo de custo mínimo não é de posto completo (a soma de suas linhas é o vetor nulo), entretanto, pode-se adicionar um vetor unitário $e_m = (0, ..., 1)^T \in R^m$, de modo que a matriz $[A|e_m]$ torna-se de posto completo; neste caso, denominamos este nó de raiz, da mesma forma que e_m é denominado arco raiz [11].

No problema de fluxo de custo mínimo, é possível mostrar que cada base é uma matriz triangular [11, 58]. Também é possível mostrar que existe uma correspondência biunívoca entre bases triangulares de A e árvores enraizadas do grafo [58]. A estrutura destas bases permite desenvolver algoritmos especializados que aproveitam as particularidades estruturais de B e A_s para resolver de modo eficiente os sistemas,

$$\begin{array}{rcll} y^T B & = & c_B^T & [Cálculo \ das \ variáveis \ duais] \\ B\hat{A}_s & = & A_s & [Atualizacão \ da \ coluna \ s] \end{array} \qquad (1.3)$$

Exemplo 1 (continuação)

Para facilitar a apresentação vamos adotar a notação x_{ij}, z_{ij}, c_{ij} para representar os dados associados ao arco (i, j). No seguinte exemplo, é feito uma iteração completa do

método simplex especializado para redes onde são analisados os passos mais importantes para ilustrar o processo de solução do problema de fluxo de custo mínimo e mostrar como são resolvidos os sistemas definidos em (1.3). Além disto, é apresentado a relação que existe entre resolução de sistemas triangulares e árvores enraizadas associadas à base B.

Considere a base B formada pelos arcos $(2,3)$, $(3,4)$, $(4,5)$, $(1,5)$ e o arco raiz da Figura 1.2.

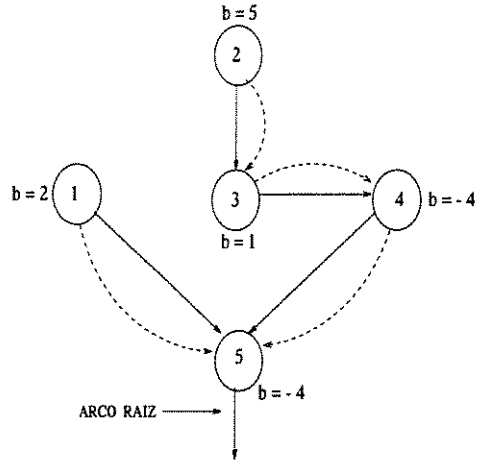


Figura 1.2: Árvore enraizada associada à solução inicial

a) Cálculo das variáveis básicas ($Bx_B = b$)

$$\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & -1 & 1 & & \\ & & -1 & 1 & \\ -1 & & & -1 & 1 \end{pmatrix} \begin{pmatrix} x_{15} \\ x_{23} \\ x_{34} \\ x_{45} \\ x_5 \end{pmatrix} = \begin{pmatrix} 2 \\ 5 \\ 1 \\ -4 \\ -4 \end{pmatrix}$$

A solução do sistema triangular $Bx_B = b$ é obtida diretamente, percorrendo a árvore enraizada das folhas à raiz (vide figura 1.2). Usando o vetor b e *reordenando* as

equações de conservação de fluxo nos nós, temos:

$$\begin{array}{lrcrcl}
\textit{nó } 1: & x_{15} & = & b_1 = & 2 & \Rightarrow \quad x_{15} = 2 \\
\textit{nó } 2: & x_{23} & = & b_2 = & 5 & \Rightarrow \quad x_{23} = 5 \\
\textit{nó } 3: & x_{34} - x_{23} & = & b_3 = & 1 & \Rightarrow \quad x_{34} = 6 \\
\textit{nó } 4: & x_{45} - x_{34} & = & b_4 = & -4 & \Rightarrow \quad x_{45} = 2 \\
\textit{nó } 5: & x_5 - (x_{45} + x_{15}) & = & b_5 = & -4 & \Rightarrow \quad x_5 = 0
\end{array}$$

b) Cálculo das variáveis duais ($y^T B = c_B^T$)

$$\begin{pmatrix} y_1 & y_2 & y_3 & y_4 & y_5 \end{pmatrix} \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & -1 & 1 & & \\ & & -1 & 1 & \\ -1 & & & -1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & -4 & 0 & 3 & 0 \end{pmatrix}$$

A solução do sistema triangular $y^T B = c_B^T$ é obtido diretamente percorrendo a árvore enraizada da raiz às folhas (vide Figura 1.3). Usando o vetor c_B e *reordenando* as equações de custo relativo nulo nos arcos básicos, temos:

$$\begin{array}{lrcrcl}
\textit{arco } 5 & : \; y_5 = 0 & & & \Rightarrow & y_5 = 0 \\
\textit{arco } (1,5) & : \; y_1 - y_5 & = & c_{15} = 2 & \Rightarrow & y_1 = 2 \\
\textit{arco } (4,5) & : \; y_4 - y_5 & = & c_{45} = 3 & \Rightarrow & y_4 = 3 \\
\textit{arco } (3,4) & : \; y_3 - y_4 & = & c_{34} = 0 & \Rightarrow & y_3 = 3 \\
\textit{arco } (2,3) & : \; y_2 - y_3 & = & c_{23} = -4 & \Rightarrow & y_2 = -1
\end{array}$$

A árvore enraizada associada a este sistema está representada na Figura 1.3.

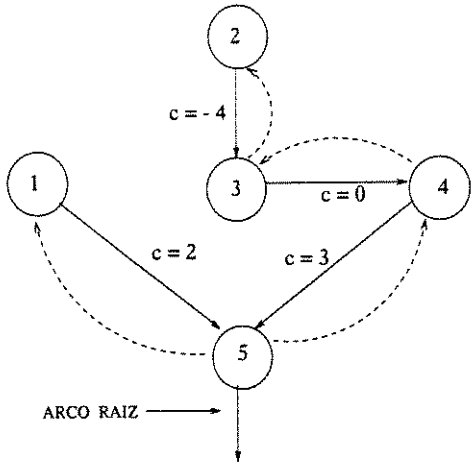


Figura 1.3: Cálculo das variáveis duais

c) Cálculo dos custos relativos ($z = c - A^T y$)

Usando-se os valores das variáveis duais y, calcula-se o valor de z_{ij}. Para calcular z_{ij} $\forall (i,j) \in N$ (para todos os arcos não básicos), aplica-se a relação:

$$z_{ij} = c_{ij} - y_i + y_j$$

d) Determinação da variável que sai e pivoteamento

No exemplo $z_{13} = -1 < 0$. Logo, x_{13} é um candidato para entrar na base, e deve melhorar o valor da função objetivo. Devemos aumentar o valor da variável x_{13}, ajustando convenientemente as variáveis básicas, de modo a manter a factibilidade e determinar a primeira variável básica que se anula. Neste exemplo, a direção de deslocamento Δx é definida por,

$$\begin{aligned}
\Delta x_{23} &= \ \ 0 \quad (\hat{A}_{23,13} = \ \ 0) \\
\Delta x_{15} &= -1 \quad (\hat{A}_{15,13} = +1) \\
\Delta x_{34} &= +1 \quad (\hat{A}_{34,13} = -1) \\
\Delta x_{45} &= +1 \quad (\hat{A}_{45,13} = -1) \\
\Delta x_{13} &= +1 \\
\Delta x_{ij} &= \ \ 0 \quad \text{para os demais arcos}
\end{aligned}$$

o que define um ciclo no grafo (Figura 1.4). Este ciclo é formado pelo arco x_{13} e por alguns arcos básicos x_{34}, x_{45}, x_{15} (este último com sentido contrário). A aplicação do teste de razão indica que x_{15} deve ser escolhido para sair da base com $\alpha = 2 = 2/1$.

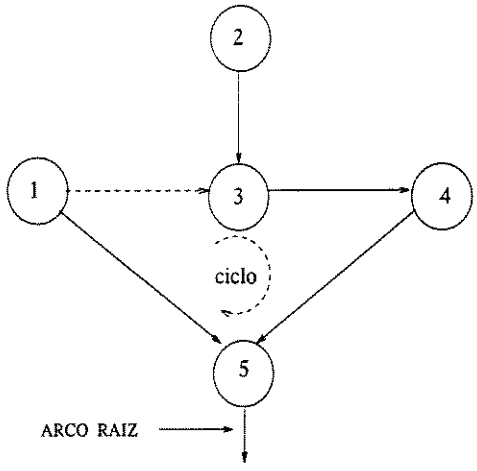


Figura 1.4: Ciclo para determinar a variável que sai

e) Atualização da árvore (B^{-1})

É obtida eliminando o arco $(1,5)$ e incorporando o arco $(1,3)$ (vide Figura 1.5).

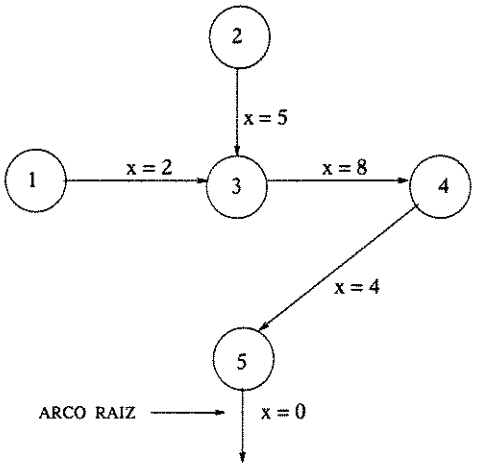


Figura 1.5: Nova solução básica factível

f) Observação. É importante salientar que a árvore geradora deve ser mantida com uma estrutura de dados apropriada para que as operações de determinação do ciclo e atualização de fluxo, preços e da árvore geradora sejam realizadas de modo eficiente.

1.4 Métodos de Pontos Interiores para Solução de Programas Lineares

Como ilustração dos métodos de pontos interiores, apresenta-se o método afim-primal. Mostra-se também algumas particularidades desses métodos na resolução de problemas de fluxo de custo mínimo.

Considere novamente o problema (1.2).

Seja o conjunto de soluções factíveis,

$$S = \{x \in R^n : Ax = b, x \geq 0\}$$

E o conjunto de soluções interiores factíveis,

$$S^0 = \{x \in S : x > 0\}$$

Os metódos de pontos interiores trabalham com pontos que estão no conjunto S^0. Ou seja, $x \in S^0$. A figura 1.6 mostra o conjunto de soluções factíveis, e uma seqüência de pontos no interior deste conjunto deslocando-se em direção à solução ótima.

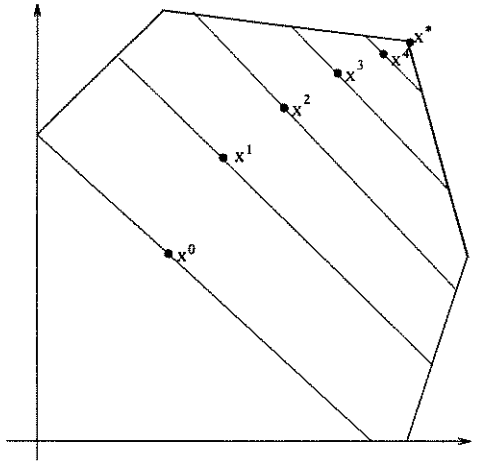


Figura 1.6: Seqüência de pontos no conjunto S^0

Os diversos métodos de pontos interiores (afins, afins com centragem, primais−duais com trajetória central), são caracterizados pelas direções de deslocamento no conjunto S^0.

Para ilustrar como são calculadas essas direções de deslocamento, apresentaremos o método afim-primal.

Dado um ponto interior factível x, define-se:

- Uma estimativa das variáveis duais (y): $y = (AX^2A^T)^{-1}AX^2c$
- Custos relativos (z): $z = c - A^Ty$
- Direção de movimento (Δx): $\Delta x = -X^2z$

onde $X = diag(x)$ é a matriz de escalamento.

O método afim−primal é resumido na seqüência de passos a seguir.

Seja $x \in S^0$
Repita

X	$\leftarrow$	$diag(x)$	[Matriz diagonal]
y	$\leftarrow$	$(AX^2A^T)^{-1}AX^2c$	[Cálculo de y]
z	$\leftarrow$	$c - A^Ty$	[Custos relativos]
Δx	$\leftarrow$	$-X^2z$	[Direção de movimento]
α	$\leftarrow$	$min[-x_j/\Delta x_j : \Delta x_j < 0]$	[Teste de razão]
x	$\leftarrow$	$x + \beta\alpha\Delta x$	[Novo ponto]

Até $(|x^Tz| < \epsilon)$

onde β é um parâmetro no intervalo $(0, 1)$, usado para garantir que o novo ponto é ainda um ponto interior, e ϵ é uma tolerância pré-estabelecida.

Seja,

$$Q = AX^2A^T \quad e \quad \hat{b} = AX^2c$$

O maior esforço computacional do método afim-primal está na construção da matriz Q e no cálculo da estimativa das variáveis duais y (resolução do sistema $Qy = \hat{b}$).

É possível mostrar que a matriz Q é uma matriz simétrica e positiva definida. Isto é:

$$\begin{aligned} Q &= Q^T \\ x^TQx &> 0 \quad \forall x \neq 0 \quad (x \in R^n) \end{aligned}$$

A solução do sistema $Qy = \hat{b}$ é realizada com métodos diretos (normalmente usando a fatoração de Choleski) ou métodos iterativos (normalmente usando gradiente conjugado pré-condicionado).

1.4.1 Pontos Interiores em Rede

Uma característica importante dos problemas de fluxo de custo mínimo que é explorada nos métodos de pontos interiores está relacionada com a matriz $Q = AX^2A^T$. Para que esta matriz seja realmente (simétrica) positiva definida é necessário que a matriz A seja de posto completo. Isto implica que para uma rede conectada é necessário eliminar (qualquer) uma das linhas da matriz de incidência o que significa que a matriz Q é uma $(m-1) \times (m-1)$–matriz quadrada.

O maior esforço computacional dos métodos de pontos interiores a cada iteração, é composto pela construção da matriz Q (complexidade $O(m^2n)$) e a solução do sistema $Qy = \hat{b}$ (complexidade $O(m^3)$). No caso de um Problema de Fluxo de Custo Mínimo a construção da matriz Q é realizada com complexidade $O(n)$ que corresponde a percorrer os arcos da rede.

Finalmente, a resolução do sistema $Qy = \hat{b}$ feita com o método gradiente conjugado pré-condicionado utilizando o pré-condicionamento diagonal é efetuado com complexidade $O(n)$ por iteração sem utilizar a matriz Q explicitamente (não requer a construção desta matriz). Além disto, o pré-condicionamento com a árvore geradora máxima também é feito de modo eficiente (vide capítulo 3).

O algoritmo abaixo é utilizado para determinar o produto $u = Qv$ sem construir a matriz Q.

$u[1...m] \leftarrow [0...0]$
`for` $k \leftarrow 1..n$ `do`
    $u[t_k] \leftarrow u[t_k] + x_k^2 * (v[t_k] - v[h_k])$
    $u[h_k] \leftarrow u[h_k] + x_k^2 * (v[h_k] - v[t_k])$

onde o arco $k = (t_k, h_k)$ apresenta origem t_k e destino h_k.

Exemplo (continuação)

Eliminando a última linha (nó 5) da matriz de incidência, pode-se determinar Q da seguinte forma,

$$Q = \begin{pmatrix} x_{12}^2 + x_{13}^2 + x_{15}^2 & -x_{12}^2 & -x_{13}^2 & 0 \\ -x_{12}^2 & x_{12}^2 + x_{23}^2 + x_{42}^2 & -x_{23}^2 & -x_{42}^2 \\ -x_{13}^2 & -x_{23}^2 & x_{13}^2 + x_{23}^2 + x_{34}^2 + x_{53}^2 & -x_{34}^2 \\ 0 & -x_{42}^2 & -x_{34}^2 & x_{42}^2 + x_{34}^2 + x_{45}^2 \end{pmatrix}$$

cuja diagonal é composta pela soma dos quadrados do fluxo dos arcos que incidem (com origem ou destino) no nó e cujos elementos não-diagonais é o quadrado do fluxo do arco que liga os nós, se existir.

Considere o fluxo factível x, representado na Figura 1.7.

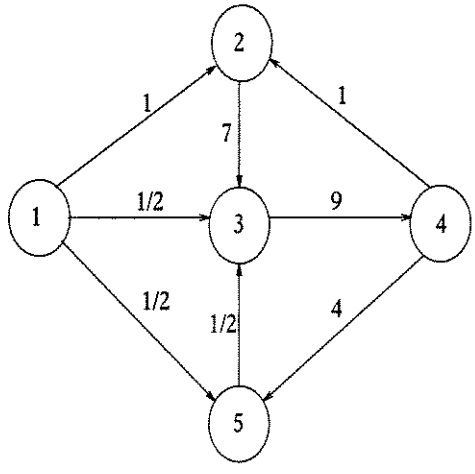


Figura 1.7: Fluxo factível (vetor x)

a matriz associada é:

$$Q = \begin{pmatrix} \frac{3}{2} & -1 & \frac{-1}{4} & 0 \\ -1 & 51 & -49 & -1 \\ \frac{-1}{4} & -49 & \frac{261}{2} & -81 \\ 0 & -1 & -81 & 98 \end{pmatrix}$$

1.5 Histórico

Ao realizar trabalhos para a Força Aérea dos Estados Unidos em 1947, George B. Dantzig identificou em vários problemas de programação e planejamento militar, uma caraterística comum: a tarefa puramente matemática de maximização ou minimização de uma função linear cujas variáveis são restritas a valores satisfazendo um sistema de restrições lineares, isto é, um conjunto de equações e/ou inequações lineares. Foi natural Tjallin C. Koopmans sugerir para o assunto o nome de programação linear, em vista do primeiro artigo publicado por Dantzig: "Programming in a Linear Structure", publicado na revista Econometrica [24].

Conta a história que o famoso matemático Fourier foi provalmente o primeiro a estudar sistematicamente desigualdades lineares e a mostrar sua importância em mecânica e teoria da probabilidade. Ele estava interessado em achar um ajuste com o menor desvio máximo para um sistema de equações lineares, que ele reduziu ao problema de achar o ponto de um poliedro com valor mínimo (este é provavelmente o exemplo mais antigo de um problema de programação linear). A proposta de solução de Fourier foi a de percorrer vértice por vértice, descendo para o valor mínimo. Observe que isto constitui a essência do método simplex. Esse método foi desenvolvido por Dantzig em 1947 e foi durante muitos anos reconhecido como o método mais eficiente para trabalhar com problemas de programação linear. De acordo com Dantzig, o nome método simplex foi sugerido por T. S. Motzkin.

Devido ao crescente número de aplicações, o método simplex vem sendo desde a sua criação objeto de inúmeras pesquisas teóricas e computacionais. Em outubro de 1947, John von Newmann, trocando idéias com Dantzig, introduziu o conceito de problema de programação linear dual, que está associado a cada problema de programação linear original (ou primal). Subsequentemente, Gale, Kuhn e Tucker [38] formularam e provaram o Teorema da Dualidade.

Em 1954, C. Lemke desenvolveu o algoritmo dual simplex [63], como uma variação do primal simplex padrão. Uma outra variante é o método primal-dual, desenvolvido por Dantzig, Ford e Fulkerson [26].

Outra alternativa para otimização em redes é o método out-of-kilter de Fulkerson

[36], posteriormente interpretado como um método primal–dual.

Outros trabalhos contribuiram para desenvolver áreas de programação linear com estruturas particulares, tais como a programação em redes [37]. A programação em redes trata de uma classe de problemas que explora a estrutura especial onde cada coluna da matriz de restrições possui somente dois elementos não nulos: $+1$ e -1. Nesta classe de problemas destacam-se os problemas de fluxo de custo mínimo, objeto de estudo deste trabalho, e suas especializações; como problemas de caminho mínimo, designação, transporte, transbordo e circulação; envolve também suas ramificações, como problemas de fluxos generalizados e multifluxos.

O método simplex especializado para redes, conduz a grandes economias de memória, e tempo de processamento. Seu desenvolvimento, iniciado com o trabalho de Dantzig foi formalizado no artigo de Ellis Johnson [54].

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 contribuicões à teoria de distribuição ótima de recursos. Embora estes pesquisadores tenham investigado uma grande variedade de problemas de otimização, ambos estão associados com algumas das primeiras publicações descrevendo problemas de fluxos em redes como 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 de fluxo de carga entre as diferentes rotas da rede da estrada de ferro de tal maneira que satisfaça às restricões de capacidade nas rotas enquanto minimiza o gasto de combustível.

Embora na prática o algoritmo simplex seja eficiente, pode-se construir classes de problemas de programação linear mostrando que o comportamento desse algoritmo no pior caso não é polinomial, e sim exponencial [82, 84]. 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 aperfeiçoamentos do método simplex podem ser encontrados em [45, 53], por exemplo. Entretanto, o esforço computacional permanece não polinomial. Com o intuito de reduzir este esforço computacional, tem sido propostos métodos conceitualmente diferentes do simplex. Em 1979 o matemático russo

Leonid G. Khachiyan, mostrou que o método para otimização convexa desenvolvido pelo matemático russo N. Z. Shor e Gershovich [91] pode ser adaptado para produzir o método dos elipsóides [57, 18] para resolução de problemas de programação linear com complexidade $O(nL)$, onde L é o tamanho dos dados do problema em *bits*.

Em 1984, N. Karmarkar divulgou o Método Projetivo [55], 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 proposto utilizava um formato especial para o problema e hipóteses restritivas. Vários pesquisadores introduziram melhoramentos no método e desenvolveram variantes para o problema em formato padrão. Entre outros, Anstreicher [4], Gay [39], De Ghellinck e Vial [28], Gonzaga [47], Todd e Burrel [93], e Ye [99]. Uma simplificação radical do algoritmo proposto por Karmarkar gerou o método afim-escala [1]. Embora não fosse conhecido no ocidente, esse algoritmo já existia havia vinte e nove anos, publicado por Dikin na USSR [30]. Na sua forma original o método afim-escala é não polinomial, mas seus resultados computacionais normalmente superiores ao simplex muito contribuiram para o desenvolvimento dos métodos de pontos interiores.

Resultados essencialmente novos foram obtidos a partir de 1986, com o estudo de métodos de trajetória central. A trajetória central foi inicialmente estudada por Bayer e Lagarias [13] e por Megiddo [74]. Renegar [86] 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 [95], seguindo a metodologia de Renegar, reduziu o limite obtido por Karmarkar para $O(n^3L)$ operações. Por um caminho independente, Gonzaga [47] obteve o mesmo limite simultaneamente, utilizando um algoritmo do tipo penalidades.

Novos algoritmos foram propostos a partir do conceito de trajetória central, incluindo os de Koyima, Mizuno e Yoshise [62], e de Monteiro e Adler [79]. Fizeram-se também extensões para programação quadrática convexa, obtendo-se os mesmos limites de complexidade. A partir de 1989 surgiram algoritmos que procuram acompanhar a trajetória central por passos longos [52], ao invés dos passos curtos, essenciais para as demonstrações

de complexidade da ordem de $\sqrt{n}L$ nos primeiros métodos de trajetória central.

Aplicações de algoritmos de pontos interiores para solução de problemas de fluxo de custo mínimo surgiram a partir de 1993 [88]. O trabalho de Resende e Veiga [88] resolve problemas de designação usando o método afim-dual. Na época, somente conseguia competir com o método simplex para problemas de grande porte (4.10^4 nós e 2.10^6 arcos) e, usando processamento paralelo.

Uma outra implementação de métodos de pontos interiores para resolver problemas de fluxos em redes, foi publicada em (1993) por Ramakrishnan, Karmarkar e Kamath [85]. Esta implementação foi usada para resolver problemas de designação com até 32768 nós e foi comparado com o código AUCTION de Bertsekas e Kastañon [16].

Posteriormente Resende e Veiga obtiveram melhorias significativas com uma implementação baseada no gradiente conjugado pré-condicionado, usando pré-condicionadores diagonal e árvore geradora máxima para resolver problemas de fluxo de custo mínimo de grande porte [89]. Eles resolveram 250 problemas com redes de 256 até 262144 nós, e fizeram comparações com os códigos NETFLO, RELAXT-3 e CPLEX. Os métodos de pontos interiores mostraram bom desempenho com relação a esses três códigos.

1.6 Apresentação do Trabalho

Este trabalho faz um estudo dos diversos métodos de pontos interiores e propõe alternativas de implementações para resolver problemas de fluxo de custo mínimo. No capítulo 2, é feita uma apresentação dos principais métodos de pontos interiores, dando-se ênfase aos algoritmos mais importantes do ponto de vista de implementações computacionais bem sucedidas. No capítulo 3, são estudados diferentes métodos para solução de sistemas de equações do tipo $(AD^*A^T)y = \hat{b}$, onde A é a matriz de incidência nó-arco associada a um problema de fluxo de custo mínimo. Discute-se também questões relacionadas com o uso de estrutura de dados para armazenar eficientemente as informações deste sistema. O capítulo 4, apresenta o problema de falta de pontos interiores e degenerescência. No capítulo 5, são apresentados os resultados computacionais comparando os métodos implementados com o método simplex. No capítulo 6, são apresentadas extensões dos métodos de pontos interiores para a solução de problemas de fluxos generalizados.

Capítulo 2

Métodos de Pontos Interiores

2.1 Introdução

Discute-se neste capítulo as principais idéias para solução de problemas lineares de otimização através de métodos de pontos interiores. Esses métodos podem ser classificados em três grandes grupos:

- algoritmos projetivos;
- algoritmos afim−escala;
- algoritmos primais−duais.

Os métodos de pontos interiores ganharam notoriedade com a divulgação do algoritmo projetivo de Karmarkar [55]. Embora os algoritmos projetivos apresentem complexidade polinomial $O(nL)$ (Anstreicher [4, 5], De Ghellinck e Vial [28], Gonzaga [49], Karmarkar [55], Todd e Burrel [93], Ye e Kojima [100]), sob análise do pior caso, êles não têm levado a implementações com bom desempenho computacional (Tomlin [94]).

Os algoritmos afim escala, embora geralmente não tenham complexidade polinomial (quando não incluem procedimentos de *centragem*), têm exibido bom comportamento computacional (Adler e outros [1], Barnes [8], Dikin [30, 31], Vandervei e outros [97], Marsten e outros [71]). De fato, foram implementações baseadas nessas idéias que deram credibilidade à hipótese de que os métodos de pontos interiores poderiam superar o método

simplex (Adler e outros [1], Monma e Morton [78], Marsten e outros [71]). Curiosamente, verificou-se que suas bases haviam sido propostas por Dikin [30], em trabalho publicado na União Soviética, bem antes da divulgação do trabalho de Karmarkar [55].

Atualmente, a maior parte das implementações bem sucedidas têm sido baseadas em algoritmos primais duais (Gonzaga [49, 52], Kojima e outros [62], Lustig [66], Lustig e outros [67, 68, 69, 70], Megiddo [74], Mehrotra [77], Monteiro e Adler [79], Monteiro e outros [81]). Esses métodos têm ainda propriedades de convergência polinomial $O(\sqrt{n}L)$, melhores do que as do algoritmo projetivo.

Nos próximos itens, 2.2 e 2.3, apresenta-se as principais idéias dos métodos afins e primais duais, usados no desenvolvimento de algoritmos para otimização de fluxos em redes, discutidos neste trabalho. Segue-se uma interpretação desses métodos sob o enfoque do método de Newton. O item 2.5 discute implementações de algoritmos de pontos interiores com variáveis canalizadas. A parte final do capítulo é dedicada a inicializações e comentários adicionais, abordados, respectivamente, nos itens 2.6 e 2.7.

2.2 Métodos Afins

Os métodos afins têm duas grandes variantes, primal e dual, ambas fundamentadas na minimização dos desvios da condição de folgas complementares (Chandru e Kochar [22]).

2.2.1 Método Afim Primal

Considere o problema de programação linear sem variáveis canalizadas, P,

$$(P)\begin{cases} Minimizar & c^T x \\ Sujeito\ a & Ax = b \\ & x \geq 0 \end{cases}$$

e seu dual, D,

$$(D)\begin{cases} Maximizar & b^T y \\ Sujeito\ a & A^T y \leq c \\ & y\ Irrestrito \end{cases}$$

onde $A(m \times n)$ é uma matriz de posto completo.

Pelo **Teorema das Folgas Complementares** (Luenberger [64]), se existirem dois pontos, x^* e y^*, soluções factíveis de P e D, tais que

$$X^*(c - A^T y^*) = 0$$

onde X^* é a matriz diagonal cujos elementos são as componentes do vetor x^* ($X^* = diag(x^*)$),

então x^* é solução ótima do primal e y^* é solução ótima do dual.

Seja x um ponto interior factível, isto é, $x > 0$ e $Ax = b$, e $\gamma(y)$ o vetor dos desvios da condição de folgas complementares,

$$\gamma(y) = X(c - A^T y)$$

onde $X = diag(x)$.

Para tentar atender a condição de folgas complementares, pode-se obter o vetor de variáveis $\hat{y}$ que minimizem o quadrado do desvio $\gamma(y)$. Para isso, resolve-se o problema quadrático irrestrito a seguir.

$$Minimizar\ \ f(y) = \gamma^T(y)\gamma(y)$$

onde f: $R^m \longmapsto R$ é uma função quadrática, convexa e não negativa.

O gradiente de $f(y)$ é dado por:

$$\nabla f(y) = 2(-XA^T)^T(Xc - XA^T y)$$

No ponto ótimo,

$$\nabla_y f(y) = 0 \quad \Rightarrow \quad -AX^2(c - A^T y) = 0 \quad \Rightarrow \quad (AX^2A^T)y = AX^2c$$

$$\hat{y} = (AX^2A^T)^{-1}AX^2c$$

Portanto, se a condição de folgas complementares estiver satisfeita para $\hat{y}$, êle é o vetor de variáveis duais, solução do problema D e o ponto $\hat{x}$ associado é solução de P. Caso contrário, se a condição de folgas complementares não estiver atentida, $\hat{y}$ pode ser interpretado como uma estimativa das variáveis duais.

A partir de $\hat{y}$, calcula-se a estimativa do vetor de custos relativos, $\bar{c} = c - A^T\hat{y}$, e uma direção factível de descida, definida como $\Delta x = -X^2\bar{c}$.

Para mostrar que Δx é, de fato, uma direção factível de descida, demonstra-se o teorema a seguir.

TEOREMA 1 Δx é uma direção factível de descida, isto é, satisfaz as condições:

$A)$ $A\Delta x = 0$ *(a direcão é factível em x interior)*

$B)$ $c^T\Delta x < 0$ *(a direcão é de descida)*

PROVA A

$$\begin{aligned} A\Delta x &= A[-X^2\bar{c}] = A[-X^2(c - A^Ty)] = \tfrac{1}{2}\nabla f(y) \\ &= 0 \end{aligned}$$

PROVA B

$$\begin{aligned} f(y) &= [X(c - A^Ty)]^T[X(c - A^Ty)] \\ &= (c - A^Ty)^T X^2\bar{c} \\ &= (c^T - y^TA)(-\Delta x) \\ &= -c^T\Delta x + y^TA\Delta x \\ &= -c^T\Delta x > 0 \qquad\qquad [f(y) > 0] \end{aligned}$$

Portanto,

$$c^T\Delta x < 0$$

Para encontrar o novo ponto, deve-se calcular o tamanho máximo de passo, dado pela primeira componente do vetor x a se anular, isto é:

$$\alpha = min[-x_j/\Delta x_j : \Delta x_j < 0]$$

Tendo em vista os conceitos acima, o algoritmo afim primal pode ser resumido na seqüência de passos a seguir.

Dado $x = (x_j)$ um ponto inicial interior factível para P, $\beta = 0,995$ e $\epsilon = 10^{-8}$

$k \leftarrow 0$
Repita
$X \longleftarrow diag(x)$
$y \longleftarrow (AX^2A^T)^{-1}AX^2c$
$\bar{c} \longleftarrow c - A^Ty$
$\Delta x \longleftarrow -X^2\bar{c}$
$\alpha \longleftarrow \beta \ Min[-x_j/\Delta x_j : \Delta x_j < 0]$
$x \longleftarrow x + \alpha \ \Delta x$
$k \longleftarrow k + 1$
Até $(|x^T\bar{c}| < \epsilon)$

2.2.2 Método Afim Dual

Seja o seguinte problema dual equivalente D',

$$(D') \begin{cases} Maximizar & b^Ty \\ Sujeito\ a & A^Ty + z = c \\ & z \geq 0 \end{cases}$$

onde z é o vetor de variáveis de folga associadas às restrições do dual.

Considerando o problema dual equivalente D', o **Teorema das Folgas Complementares** (Luenberger [64]) estabelece que, se existirem três pontos, x^*, y^* e z^*, soluções factíveis de P e D', satisfazendo a condição

$$Z^*x^* = 0$$

onde $Z^* = diag(z^*)$,
então x^* é solução ótima do primal e (y^*, z^*) é solução ótima do dual.

Seja (y, z) um ponto interior dual factível, isto é, y é tal que $A^T y + z = c$, $z > 0$, e $\pi(x)$ o vetor dos desvios da condição de folgas complementares. Ou seja,

$$\pi(x) = Zx$$

onde $Z = diag(z)$, $z = c - A^T y$.

Na tentativa de atender a condição de folgas complementares, pode-se obter o vetor de variáveis $\hat{x}$ que minimize o quadrado do desvio $\pi(x)$, restrito a $Ax = b$.

De forma análoga ao caso primal discutido no item anterior, resolve-se o seguinte problema quadrático restrito:

$$\begin{array}{ll} Minimizar & g(x) = (Zx)^T(Zx) = \pi^T(x)\ \pi(x) \\ Sujeito\ a & Ax = b \end{array} \tag{2.1}$$

como g ($g : R^n \longrightarrow R$) é uma função quadrática, convexa e não negativa, a solução de (2.1) pode ser obtida pelas condições de estacionariedade do Lagrangeano $L(x, \lambda)$,

$$L(x, \lambda) = g(x) + \lambda^T(Ax - b)$$

As condições de otimalidade para (2.1) (Luenberger [64]), são:

$$\left\{ \begin{array}{rcl} \nabla_x L(x,\lambda) & = & 0 \\ \nabla_\lambda L(x,\lambda) & = & 0 \end{array} \right. \quad ou \quad \left\{ \begin{array}{rcl} \nabla g(x) + A^T\lambda = & 2Z^2x + A^T\lambda = & 0 \\ Ax - b = & 0 \quad \Rightarrow Ax = & b \end{array} \right. \tag{2.2}$$

Pré-multiplicando a primeira equação de (2.2) por $\frac{1}{2}Z^{-2}$, obtemos:

$$x + \frac{1}{2}Z^{-2}A^T\lambda = 0 \quad \Rightarrow \quad x = -\frac{1}{2}Z^{-2}A^T\lambda \tag{2.3}$$

Substituindo a expressão (2.3) em $Ax = b$, obtemos:

$$-\frac{1}{2}(AZ^{-2}A^T)\lambda = b$$

Logo,

$$\lambda = -2(AZ^{-2}A^T)^{-1}b$$

Portanto,

$$\hat{x} = Z^{-2}A^T(AZ^{-2}A^T)^{-1}b = -\frac{1}{2}Z^{-2}A^T\lambda$$

Observe que se verifica a restrição $Ax = b$.

A partir de um ponto y, dual factível, pode-se calcular:

- as variáveis de folga, $z = c - A^T y$ $[z = \bar{c}]$,
- a direção de movimento $(\Delta y, \Delta z)$,

$$\begin{aligned} \Delta y &= (AZ^{-2}A^T)^{-1}b \qquad [\Delta y = -\lambda/2] \\ \Delta z &= -A^T\Delta y, \end{aligned}$$

- e as variáveis primais, $x = Z^{-2}A^T\Delta y = -Z^{-2}\Delta z$

Para mostrar que $(\Delta y, \Delta z)$ é uma direção factível de subida, demonstra-se o teorema 2.

TEOREMA 2 $(\Delta y, \Delta z)$ é uma direção factível de subida em (y, z) interior, isto é, as condições $A)$, $B)$ e $C)$, abaixo, são verdadeiras.

$$\begin{aligned} A) \quad A^T\Delta y + \Delta z &= 0 \\ B) \quad b^T\Delta y &> 0 \\ C) \quad Ax &= b \end{aligned}$$

PROVA A

$$\begin{aligned} A^T\Delta y + \Delta z &= A^T(AZ^{-2}A^T)^{-1}b - A^T(AZ^{-2}A^T)^{-1}b \\ &= -\tfrac{1}{2}\nabla L_x(x, \lambda) \\ &= 0 \end{aligned}$$

$$
\begin{aligned}
g(x) &= (Zx)^T(Zx) = (Z^{-1}A^T\Delta y)^T(Z^{-1}A^T\Delta y) \\
&= (Z^{-1}A^T(AZ^{-2}A^T)^{-1}b)^T(Z^{-1}A^T\Delta y) \\
&= b^T(AZ^{-2}A^T)^{-1}AZ^{-1}(Z^{-1}A^T\Delta y) \\
&= b^T(AZ^{-2}A^T)^{-1}(AZ^{-2}A^T)\Delta y \\
&= b^T\Delta y > 0 \qquad\qquad [g(x) > 0]
\end{aligned}
$$

PROVA B

PROVA C

$$\nabla_\lambda L(x,\lambda) = Ax - b = 0$$

Portanto,

$$Ax = b$$

Tendo em vista os conceitos acima, o algoritmo afim dual pode ser resumido na seqüência de passos a seguir.

Dado (y, z) um ponto inicial interior factível para D', $\beta = 0,995$ e $\epsilon = 10^{-8}$

$k \longleftarrow 0$

Repita

$z \longleftarrow c - A^T y$

$Z \longleftarrow diag(z)$

$\Delta y \longleftarrow (AZ^{-2}A^T)^{-1}b$

$x \longleftarrow Z^{-2}A^T\Delta y$

$\Delta z \longleftarrow -A^T\Delta y$

$\alpha \longleftarrow \beta\ Min[-z_j/\Delta z_j : \Delta z_j < 0]$

$y \longleftarrow y + \alpha\ \Delta y$

$k \longleftarrow k + 1$

Até $(|x^T z| < \epsilon)$

$x = Z^{-2}A^T\Delta y$

O vetor x (variáveis primais) pode ser calculado no final, ou em qualquer ponto do laço **Repita ... Até**.

2.2.3 O Conceito de Centragem

Considere novamente o par primal P dual D de programas lineares na forma padrão

$$(P)\ \ Minimizar\ \{\ c^T x\ :\ Ax = b,\ x \geq 0\ \}$$

$$(D)\ \ Maximizar\ \{\ b^T y\ :\ A^T y + z = c,\ z \geq 0\ \}$$

Suponha que A é uma matriz de posto completo, que os problemas P e D apresentam soluções primais ótimas e duais ótimas limitadas, respectivamente, e que existam pontos interiores. O ponto central ou centro do conjunto de soluções factíveis é o único ponto x que satisfaz

$$Minimizar\ \{\ -\sum_j \ln x_j\ :\ Ax = b,\ x > 0\ \}$$

A função $-\sum_j \ln x_j$ é denominada *função barreira.* Esta função tenta manter o ponto $x = (x_j)$ o mais afastado possível da fronteira do conjunto de soluções factíveis. A rigor, esta função é definida apenas para pontos interiores ao conjunto de soluções factíveis, pois, somente neste caso $x > 0$.

A trajetória central, definida em termos de um parâmetro σ, é o conjunto de pontos que satisfazem

$$Minimizar\ \{\ -\sum_j \ln x_j\ :\ Ax = b,\ x > 0,\ c^T x = \sigma\ \}$$

Observe que esta trajetória central passa pelo ponto central e também pela solução ótima do problema, ou pelo centro do conjunto de soluções ótimas, caso existam várias.

Vale a pena mencionar que os algoritmos afins com centragem são polinomiais (Renegar [86], Fang e Puthenpura [33]).

2.3 Métodos Primais-Duais

Os métodos primais duais, inicialmente propostos por Megiddo [74], têm diversas variantes, sendo as mais usadas o método primal-dual simples (Lustig e outros [67], Mehrotra [77]) e o método primal-dual preditor-corretor (Lustig e outros [69], Mehrotra [75, 77]).

2.3.1 Método Primal Dual Simples

O Método Primal Dual tenta resolver os problemas primal (P) e dual (D') simultaneamente. Este método pode ser facilmente obtido aplicando-se o método barreira logarítmico ao problema dual D', isto é, resolvendo a seguinte família de problemas (Monteiro e Adler [79]):

$$(D_\mu)\begin{cases} Maximizar & b^T y + \mu \sum_j \ln z_j \\ Sujeito\ a & A^T y + z = c \end{cases}$$

onde $\mu > 0$ é um parâmetro de penalização (barreira).

Associando as variáveis primais (x) às restrições de igualdade, o Lagrangeano associado ao problema D_μ é dado por:

$$L_\mu(x, y, z) = b^T y + \mu \sum_j \ln z_j - x^T (A^T y + z - c)$$

Aplicando as condições de Kuhn Tucker, para μ fixo, temos:

$$\nabla_x L(x, y, z) = 0 \quad \Rightarrow \quad c - A^T y - z \quad = 0 \tag{2.4}$$

$$\nabla_y L(x, y, z) = 0 \quad \Rightarrow \quad b - Ax \quad = 0 \tag{2.5}$$

$$\nabla_z L(x, y, z) = 0 \quad \Rightarrow \quad \mu Z^{-1} e - x \quad = 0 \tag{2.6}$$

onde $Z = diag(z)$, $e = [1...1]^T \in R^n$.

Tendo em vista que $Z = diag(z)$, (2.6) pode ser escrito na forma a seguir.

$$ZXe = \mu e, \ onde, \ X = diag(x)$$

Pode-se reescrever o sistema (2.4) – (2.6) numa forma mais usual. Assim,

$$Ax = b \tag{2.7}$$

$$A^T y + z = c \tag{2.8}$$

$$ZXe = \mu e \tag{2.9}$$

As condições (2.7) e (2.8) são as condições de factibilidade dos problemas primal P e dual D'. Quando $\mu \to 0$, (2.9) é a condição de folgas complementares.

As direções de deslocamento $(\Delta x, \Delta y, \Delta z)$ são obtidas resolvendo o seguinte sistema de equações:

$$A\Delta x = b - Ax \tag{2.10}$$

$$A^T\Delta y + \Delta z = c - A^T y - z \tag{2.11}$$

$$Z\Delta x + X\Delta z = \mu e - ZXe \tag{2.12}$$

No sistema acima, temos $m + 2n$ equações e $m + 2n$ incógnitas. Para resolvê-lo, pode-se fazer manipulações algébricas de forma a se obter um sistema de ordem m.

Pré-multiplicando a equação (2.12) por Z^{-1}, temos:

$$\Delta x + Z^{-1}X\Delta z = \mu Z^{-1}e - Xe \tag{2.13}$$

Pré-multiplicando a equação (2.13) por A, temos:

$$A\Delta x + AZ^{-1}X\Delta z = \mu AZ^{-1}e - AXe \tag{2.14}$$

Definindo $d_P = b - Ax$, $d_D = c - A^T y - z$ e usando (2.10) e (2.11), temos: $A\Delta x = d_P$ e $\Delta z = d_D - A^T\Delta y$. Logo a equação (2.14) torna-se:

$$(b - Ax) + AZ^{-1}Xd_D - (AZ^{-1}XA^T)\Delta y = \mu AZ^{-1}e - AXe \tag{2.15}$$

Portanto,

$$\Delta y = (AZ^{-1}XA^T)^{-1}(b + AZ^{-1}Xd_D - \mu AZ^{-1}e)$$

Usando (2.13), Δx é dado por:

$$\Delta x = -Z^{-1}X\Delta z - Xe + \mu Z^{-1}e$$

Em resumo, Δy, Δz, Δx são dados pelas seguintes expressões:

$$\Delta y = (AZ^{-1}XA^T)^{-1}(b + AZ^{-1}Xd_D - \mu AZ^{-1}e) \tag{2.16}$$

$$\Delta z = d_D - A^T\Delta y \tag{2.17}$$

$$\Delta x = -Z^{-1}X\Delta z - Xe + \mu Z^{-1}e \tag{2.18}$$

Nas equações (2.16), (2.17) e (2.18), pode-se observar que temos uma parcela com o parâmetro μ e outra parcela sem este parâmetro. Fazendo-se $\mu = 0$ pode-se construir um algoritmo mais simples, denominado Primal-Dual Afim Escala (Monteiro e outros [81]). Neste caso, as direções de deslocamento são dadas pelas seguintes expressões:

$$\Delta y = (AZ^{-1}XA^T)^{-1}(b + AZ^{-1}Xd_D) \tag{2.19}$$

$$\Delta z = d_D - A^T\Delta y \tag{2.20}$$

$$\Delta x = -Z^{-1}X\Delta z - Xe \tag{2.21}$$

Em princípio, o algoritmo Primal-Dual Afim Escala teve apenas interesse teórico, decorrente de suas propriedades de convergência polinomial (Monteiro e outros [81]). No entanto, êle pode ser interpretado como a fase de *predição*, do método preditor-corretor (Mehrotra [77]), discutido no próximo item. O método preditor-corretor levou a implementações muito bem sucedidas (Lustig e outros [70], Mehrotra [77]).

2.3.2 Método Preditor-Corretor

Em 1992 Mehrotra [77] propôs o método preditor-corretor. Novamente, trata-se de resolver simultaneamente, o par de problemas primal e dual (P e D'). Neste caso, as estimativas das direções de deslocamento incluem termos de segunda ordem. Para se obter essas direções, ao invés de se resolver uma família de problemas com relação ao parâmetro de centragem μ, faz-se o seguinte procedimento:

Seja $(\hat{x}, \hat{y}, \hat{z})$ a nova solução definida pelas condições a seguir.

$$\begin{aligned} \hat{x} &\leftarrow x + \Delta x \\ \hat{y} &\leftarrow y + \Delta y \\ \hat{z} &\leftarrow z + \Delta z \end{aligned}$$

Substituindo essa solução nas equações (2.7), (2.8) e (2.9), obtemos:

$$\begin{aligned} A(x+\Delta x) &= b \\ A^T(y+\Delta y)+(z+\Delta z) &= c \\ (X+\Delta X)(Z+\Delta Z)e &= \mu e \end{aligned}$$

onde $\Delta X = diag(\Delta x)$ e $\Delta Z = diag(\Delta z)$ são matrizes diagonais.

Escrevendo essas equações em termos matriciais,

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - A^T y - z \\ \mu e - XZe - \Delta X \Delta Z e \end{bmatrix} \tag{2.22}$$

Examinando a equação matricial (2.22), vemos que o lado direito da última linha é similar à equação (2.12); a diferença está no termo não linear ($\Delta X \Delta Z e$) que aparece no lado direito da equação (2.22). Este termo não é conhecido, no entanto, podemos calcular uma aproximação para o mesmo, usando o método primal-dual afim escala, isto é, aplicando o método primal-dual simples com o parâmetro de penalização $\mu = 0$. Isto corresponde à etapa de predição do método preditor-corretor proposto por Mehrotra [77]. O sistema de equações a ser resolvido nesta etapa é dado por:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} = \begin{bmatrix} d_P \\ d_D \\ -XZe \end{bmatrix} \tag{2.23}$$

onde,

$$\begin{aligned} d_P &= b - Ax \quad \text{(infactibilidade do primal)}, \\ d_D &= c - A^T y - z \quad \text{(infactibilidade do dual)}, \end{aligned}$$

e $(\delta x, \delta y, \delta z)$ é uma aproximação das direções de deslocamento $(\Delta x, \Delta y, \Delta z)$.

Fazendo manipulações algébricas, tem-se:

$$\begin{aligned} \delta y &= (AZ^{-1}XA^T)^{-1}(AXe + AZ^{-1}Xd_D + d_P) = (AZ^{-1}XA^T)^{-1}(b + AZ^{-1}Xd_D) \\ \delta z &= d_D - A^T \delta y \\ \delta x &= Z^{-1}(-XZe - X\delta z) = -Xe - Z^{-1}X\delta z \end{aligned}$$

Uma vez obtida uma aproximação do termo não linear $\Delta X \Delta Z e$, denotado por $\delta X \delta Z e$, deve-se substituir a aproximação no lado direito da equação matricial (2.22) e resolver o sistema. Esta é a etapa de correção.

O novo sistema a ser resolvido é:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - A^T y - z \\ \mu e - XZe - \delta X \delta Z e \end{bmatrix} \tag{2.24}$$

Fazendo manipulações algébricas, obtemos:

$$\begin{aligned} \Delta y &= (AZ^{-1}XA^T)^{-1}(b + AZ^{-1}Xd_D + AZ^{-1}\delta X \delta Z e - \mu AZ^{-1}e) \\ \Delta z &= d_D - A^T \Delta y \\ \Delta x &= Z^{-1}(\mu e - XZe - \delta X \delta Z e - X\Delta z) \end{aligned}$$

Em resumo, o algoritmo preditor-corretor tem duas etapas:

- **PREDIÇÃO** onde se calcula uma aproximação do termo não linear $(\Delta X \Delta Z e)$, denotado por $(\delta X \delta Z e)$, usando o método primal-dual afim escala; isto corresponde a aplicar o método primal-dual simples, apresentado na seção anterior, com o parâmetro de penalização $\mu = 0$;
- **CORREÇÃO** usando a aproximação obtida na etapa de predição, deve-se resolver o sistema de equações (2.24), para obter as direções de deslocamento $(\Delta x, \Delta y, \Delta z)$ primais-duais.

Calcula-se os passos α_P e α_D, nos espaços primal e dual, respectivamente, escolhidos de forma que o novo ponto $(x + \alpha_P \Delta x, y + \alpha_D \Delta y, z + \alpha_D \Delta z)$ seja ainda interior.

Tendo em vista os conceitos acima, o algoritmo preditor-corretor pode ser resumido na seqüência de passos a seguir.

Dado um ponto (x, y, z) tal que $x > 0$, $z = c - A^T y > 0$, $\beta = 0,9995$ e $\epsilon = 10^{-8}$

$k \longleftarrow 0$

Repita

$\mu \longleftarrow x^T z/n^2$

$\Theta \longleftarrow Z^{-1}X$

$\delta y \longleftarrow (A\Theta A^T)^{-1}(AXe + AZ^{-1}Xd_D + d_P)$

$\delta z \longleftarrow d_D - A^T \delta y$

$\delta x \longleftarrow Z^{-1}(-XZe - X\delta z)$

$\Delta y \longleftarrow (A\Theta A^T)^{-1}(b + AZ^{-1}Xd_D + AZ^{-1}X\delta X\delta Ze - \mu Z^{-1}Xe)$

$\Delta z \longleftarrow d_D - A^T \Delta y$

$\Delta x \longleftarrow Z^{-1}(\mu e - XZe - \delta X\delta Ze - X\Delta z)$

$\alpha_P \longleftarrow \beta \ min[-x_j/\Delta x_j : \Delta x_j < 0]$

$\alpha_D \longleftarrow \beta \ min[-z_j/\Delta z_j : \Delta z_j < 0]$

$x \longleftarrow x + \alpha_P \Delta x$

$y \longleftarrow y + \alpha_D \Delta y$

$z \longleftarrow z + \alpha_D \Delta z$

$k \longleftarrow k + 1$

Até $\frac{x^T z}{1+|b^T y|} < \epsilon$

Pode-se fazer algumas observações adicionais a respeito do algoritmo preditor-corretor:

- A cada iteração do algoritmo, deve-se resolver dois sistemas simétricos e definido positivos com a mesma matriz de coeficientes $A\Theta A^T$, porém com lados direitos distintos;
- Quando métodos diretos são usados para solução do sistema, antes de se entrar ao laço **Repita ... Até**, deve-se fazer a fatoração simbólica da matriz $A\Theta A^T$ (Duff e outros [32]); ela vai ser usada duas vezes a cada iteração (para predição e correção), com a mesma estrutura;
- Como o método usa informações dos termos de segunda ordem, espera-se que ele seja mais robusto do que o primal-dual, isto é, alcance a otimalidade com um número

menor de iterações do que o método primal-dual simples — isto normalmente se reflete em menor tempo de processamento para resolver o problema.

2.4 O Método de Newton e os Métodos de Pontos Interiores

Em trabalho recente Lustig, Marsten e Shanno [70] interpretam os métodos afins, afins com centragem e primais duais com trajetória central a partir de um passo do método de Newton. Nos próximos itens estende-se essa idéia, considerando-se as infactibilidades do primal e do dual, respectivamente, nos métodos afim-primal e afim-dual, com centragem. A interpretação apresentada evidencia novas formas para implementações, onde é dispensável a existência de pontos iniciais interiores factíveis.

2.4.1 Método Afim Primal com Centragem

Considere novamente o problema P,

$$(P) \quad Minimizar \ \{c^T x : Ax = b, x \geq 0\} \tag{2.25}$$

Resolver P equivale resolver a seguinte família de problemas (Monteiro e Adler [79]):

$$Minimizar \ \{c^T x - \mu \sum_j (\ln x_j) : Ax = b, x > 0\} \tag{2.26}$$

O Lagrangeano associado ao problema (2.26) é:

$$L(x, y) = c^T x - \mu \sum_j \ln x_j - y^T (Ax - b) \tag{2.27}$$

Aplicando as condições de Kuhn-Tucker, temos:

$$\nabla_x L(x, y) = 0 \Rightarrow c - \mu X^{-1} e - A^T y = 0 \tag{2.28}$$

$$\nabla_y L(x, y) = 0 \Rightarrow Ax - b = 0 \tag{2.29}$$

onde $X = diag(x_j)$ e $e = [1..1]^T \epsilon R^n$.

Para resolver os sistemas não lineares (2.28) e (2.29) podemos dar passos do método de Newton. Assim,

$$\begin{pmatrix} A & 0 \\ -\mu X^{-2} & -A^T \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} b - Ax \\ \mu X^{-1}e - c + A^T y \end{pmatrix} \tag{2.30}$$

O sistema (2.30) tem $m + n$ equações e $m + n$ variáveis. No entanto, podemos fazer manipulações algébricas para reduzir a ordem do mesmo, obtendo um sistema de ordem m.

Definindo-se a infactibilidade do primal como d_P, $d_P = b - Ax$, pode-se reescrever o sistema (2.30).

$$A\Delta x = d_P \tag{2.31}$$

$$-\mu X^{-2}\Delta x - A^T \Delta y = \mu X^{-1}e - c + A^T y \tag{2.32}$$

Pré-multiplicando (2.32) por AX^2, temos:

$$-\mu A\Delta x - (AX^2A^T)\Delta y = \mu AXe - AX^2(c - A^T y)$$

Portanto,

$$\Delta y = (AX^2A^T)^{-1}\{\mu(-d_P - Ax) + AX^2(c - A^T y)\} \tag{2.33}$$

Para calcular Δx, podemos pré-multiplicar a equação (2.32) por $-X^2$, obtendo:

$$\mu\Delta x = -X^2A^T\Delta y - \mu Xe + X^2(c - A^T y) \tag{2.34}$$

Substituindo o valor de Δy, obtido em (2.33) na equação (2.34), obtemos:

$$\Delta x = \frac{1}{\mu}\{X^2 - X^2A^T(AX^2A^T)^{-1}AX^2\}c - Xe + X^2A^T(AX^2A^T)^{-1}(d_P + Ax)$$

Usando-se a matriz de projeção no espaço transformado,

$$P = I - XA^T(AX^2A^T)^{-1}AX,$$

pode-se escrever:

$$\Delta x = \frac{1}{\mu}XPXc - XPe + X^2A^T(AX^2A^T)^{-1}d_P \tag{2.35}$$

Na expressão (2.35) temos três parcelas. A primeira $\frac{1}{\mu}XPXc$ representa a direção de otimalidade (direção afim primal). A segunda $-XPe$ representa a direção de centragem. A terceira parcela representa a direção de factibilidade. Se $d_P = 0$, isto é, se tivermos

factibilidade primal, pode-se fazer a interpretação apresentada por Lustig, Marsten e Shanno [70]. A direção de deslocamento Δx, torna-se:

$$\Delta x = \frac{1}{\mu} XPXc - XPe \tag{2.36}$$

Na expressão (2.36) temos duas parcelas. Se $\mu \to 0$, obtemos a direção afim (método afim primal).

Usando-se a equação (2.35), pode-se fazer uma nova implementação do método afim-primal (com ou sem centragem) adotada neste trabalho (Capítulo 5), onde não se requer a factibilidade primal da solução inicial. Exige-se apenas que satisfaça a condição de positividade.

2.4.2 Método Afim Dual com Centragem

Consideremos agora o dual D' do problema P, dado por:

$$(D') \quad Maximizar \ \{b^T y : A^T y + z = c, z \geq 0\} \tag{2.37}$$

Mostra-se que resolver o problema (2.37) equivale resolver a seguinte família de problemas (com $\mu \to 0$) (Kojima e outros [62], Monteiro e Adler [79]),

$$Maximizar \ \{b^T y + \mu \sum_j \ln z_j : z = c - A^T y, z > 0\} \tag{2.38}$$

Ou simplesmente,

$$Maximizar \ \{b^T y + \mu \sum_j \ln(c_j - a_j^T y)\} \tag{2.39}$$

onde a_j é a $j-$ésima coluna da matriz A.

O Lagrangeano associado ao problema (2.39) é dado por:

$$L_\mu(y) = b^T y + \mu \sum_j \ln(c_j - a_j^T y) \tag{2.40}$$

Aplicando as condições de Kuhn-Tucker, temos:

$$\nabla_y L(y) = 0 \Rightarrow b - \mu A Z^{-1} e = 0 \tag{2.41}$$

onde $Z = diag(z)$.

Para resolver o sistema não linear (2.41), podemos dar passos do método de Newton, obtendo:

$$\begin{pmatrix} A^T & I \\ 0 & \mu AZ^{-2} \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} c - A^T y - z \\ \mu AZ^{-1}e - b \end{pmatrix} \tag{2.42}$$

O sistema (2.42) tem $m + n$ equações e $m + n$ variáveis. No entanto, podemos fazer manipulações algébricas para reduzir a ordem do sistema, obtendo um sistema de ordem m.

Definindo-se a infactibilidade do dual como d_D, $d_D = c - A^T y - z$, pode-se reescrever o sistema (2.42) como:

$$A^T \Delta y + \Delta z = d_D \tag{2.43}$$

$$\mu AZ^{-2}\Delta z = \mu AZ^{-1}e - b \tag{2.44}$$

Pré-multplicando a equação (2.43) por AZ^{-2}, temos:

$$(AZ^{-2}A^T)\Delta y + AZ^{-2}\Delta z = AZ^{-2}d_D \tag{2.45}$$

Usando (2.44), vem:

$$AZ^{-2}\Delta z = -\frac{1}{\mu}b + AZ^{-1}e \tag{2.46}$$

Substituindo (2.46) na equação (2.45), obtemos a direção de deslocamento Δy.

$$\Delta y = \frac{1}{\mu}(AZ^{-2}A^T)^{-1}b - (AZ^{-2}A^T)^{-1}AZ^{-1}e + (AZ^{-2}A^T)^{-1}AZ^{-2}d_D \tag{2.47}$$

Usando (2.43), temos: $\Delta z = d_D - A^T \Delta y$

Na expressão (2.47) temos três parcelas. A primeira parcela representa a direção de otimalidade (direção afim dual). A segunda parcela representa a direção de centragem. A terceira parcela representa a direção de factibilidade. Se $d_D = 0$, isto é, se tivermos factibilidade dual temos a interpretação apresentada por Lustig Marsten e Shanno [70]. A direção de deslocamento $(\Delta y, \Delta z)$, torna-se:

$$\Delta y = \frac{1}{\mu}(AZ^{-2}A^T)^{-1}b - (AZ^{-2}A^T)^{-1}AZ^{-1}e \tag{2.48}$$

$$\Delta z = -A^T \Delta y \tag{2.49}$$

Na expressão (2.48), novamente aparecem duas parcelas. Se $\mu \rightarrow 0$, obtemos a direção afim (método afim dual).

De forma análoga ao método afim primal com centragem, usando-se a equação (2.47) pode-se fazer uma nova implementação do método afim dual (com ou sem centragem), adotada neste trabalho (Capítulo 5), onde se requer apenas a positividade da variável de folga do dual.

2.4.3 Métodos Primais Duais

Neste caso, devemos resolver simultaneamente os problemas (2.25) e (2.37). Novamente, consideremos o problema (2.26), com Lagrangeano dado pela equação (2.27). Aplicando as condições de Kuhn-Tucker, obtemos as equações (2.28) e (2.29). Da definição do dual, temos que:

$$z = c - A^T y \tag{2.50}$$

Substituindo-se o valor de z obtido em (2.50) no sistema (2.28), obtemos:

$$XZe = \mu e \tag{2.51}$$

No método primal-dual, deve-se resolver o sistema não linear formado pelas equações (2.29), (2.50) e (2.51). As direções de deslocamento $(\Delta x, \Delta y, \Delta z)$ são obtidas resolvendo o seguinte sistema de equações:

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} d_P \\ d_D \\ \mu e - XZe \end{pmatrix} \tag{2.52}$$

O sistema (2.52) tem $m + 2n$ equações e $m + 2n$ variáveis. No entanto, pode-se fazer manipulações algébricas para reduzir a ordem do sistema, obtendo um sistema de ordem m (Lustig e outros [67]).

$$\Delta y = (AZ^{-1}XA^T)^{-1}\{Ax + d_P + AZ^{-1}Xd_D - \mu AZ^{-1}e\} \tag{2.53}$$

$$\Delta z = d_D - A^T \Delta y \tag{2.54}$$

$$\Delta x = -\{x + Z^{-1}X\Delta z\} + \mu Z^{-1}e \tag{2.55}$$

Na expressão (2.53), temos que a direção de deslocamento Δy é uma combinação linear de quatro direções. A primeira parcela representa a direção afim, a segunda parcela representa a infactibilidade do primal, a terceira parcela representa a infactibilidade do dual e a quarta parcela representa a direção de centragem.

Com relação ao método preditor-corretor, a sua apresentação (vide item 2.3.2), já inclui essa interpretação (Lustig e outros [70], Mehrotra [77]).

2.5 Métodos de Pontos Interiores com Variáveis Canalizadas

2.5.1 Método Afim Primal

Nesta seção nós estamos interessados em resolver através do método afim primal o problema de programação linear na forma padrão com variáveis canalizadas,

$$\begin{array}{lrcl} Minimizar & c^T x & & \\ Sujeito\ a & Ax & = & b \\ & 0 \leq x & \leq & d \end{array} \tag{2.56}$$

Associando as variáveis de folga s, às restrições de canalização ($0 \leq x \leq d$). O problema (2.56) pode ser reformulado como:

$$(P_c)\begin{cases} Minimizar & c^T x & & \\ Sujeito\ a & \begin{bmatrix} A & 0 \\ I & I \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} & = & \begin{bmatrix} b \\ d \end{bmatrix} \\ & x, s & \geq & 0 \end{cases}$$

e seu dual, D_c,

$$(D_c)\begin{cases} Maximizar & b^T y + d^T v & & \\ Sujeito\ a & \begin{bmatrix} A^T & I \\ 0 & I \end{bmatrix} \begin{bmatrix} y \\ v \end{bmatrix} & \leq & \begin{bmatrix} c \\ 0 \end{bmatrix} \\ & y, v & & Irrestrito \end{cases}$$

Pelo **Teorema das Folgas Complementares** (Luenberger [64]), se existirem os pontos (x^*, s^*) e (y^*, v^*), soluções factíveis de P_c e D_c, tais que:

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

onde $X^* = diag(x^*)$ e $S^* = diag(s^*)$,

então (x^*, s^*) é solução ótima do primal e (y^*, v^*) é solução ótima do dual.

Seja $\begin{bmatrix} x \\ s \end{bmatrix}$ um ponto interior primal factível, isto é, $x > 0$, $s > 0$, $Ax = b$ e $x + s = d$, e $\Gamma(y, v)$ o vetor dos desvios da condição de folgas complementares,

$$\Gamma(y, v) = \begin{bmatrix} X(c - A^T y - v) \\ -Sv \end{bmatrix}$$

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

Para tentar atender a condição de folgas complementares, pode-se obter o vetor de variáveis $(\hat{y}, \hat{v})$ que minimizem o quadrado do desvio $\Gamma(y, v)$. Para isso, resolve-se o problema quadrático irrestrito a seguir.

$$Minimizar \;\; F(y, v) = \Gamma^T(y, v)\Gamma(y, v)$$

onde $F : R^{m+n} \longrightarrow R$ é uma função quadrática, convexa e não negativa.

Diferenciamos F com relação a y e v, obtendo:

$$\begin{aligned} \nabla_y F &= 2(-AX^2c + AX^2A^Ty + AX^2v) \\ \nabla_v F &= 2(-X^2c + X^2A^Ty + (X^2 + S^2)v) \end{aligned}$$

$$\begin{cases} \nabla_y F(y, v) = 0 \\ \nabla_v F(y, v) = 0 \end{cases} \quad ou \quad \begin{cases} (AX^2A^T)y + (AX^2)v = AX^2c \\ (X^2A^T)y + (X^2 + S^2)v = X^2c \end{cases} \tag{2.57}$$

Pré-multiplicando a segunda relação de (2.57) pela expressão $-(AX^2)(X^2 + S^2)^{-1}$, temos:

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

Somando ambos os membros, obtemos:

$$A[X^2 - X^2(X^2 + S^2)^{-1}X^2]A^T y = A[X^2 - X^2(X^2 + S^2)^{-1}X^2]c$$

Fazendo $\hat{X} = X^2 - X^2(X^2 + S^2)^{-1}X^2$, temos:

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

Portanto,

$$\hat{y} = (A\hat{X}A^T)^{-1}A\hat{X}c$$

onde,

$$\begin{aligned}
\hat{X} &= X^2 - X^2(X^2 + S^2)^{-1}X^2 \\
&= X^2[I - (X^2 + S^2)^{-1}X^2] \\
&= X^2[(X^2 + S^2)^{-1}(X^2 + S^2) - (X^2 + S^2)^{-1}X^2] \\
&= X^2(X^2 + S^2)^{-1}[X^2 + S^2 - X^2] \\
&= X^2(X^2 + S^2)^{-1}S^2
\end{aligned}$$

Portanto,

$$\hat{X} = diag(\hat{x}_j), \quad \hat{x}_j = \frac{x_j^2 s_j^2}{x_j^2 + s_j^2} \quad j = 1, ..., n \tag{2.58}$$

Se a condição de folgas complementares estiver satisfeita para $(\hat{y}, \hat{v})$, êle é o vetor de variáveis duais, solução do problema D_c e o ponto $(\hat{x}, \hat{s})$ associado é solução de P_c. Caso contrário, se a condição de folgas complementares não estiver atendida, $(\hat{y}, \hat{v})$ pode ser interpretado como uma estimativa das variáveis duais.

A partir de $\hat{y}$, calcula-se a estimativa do vetor de custos relativos, $\bar{c} = c - A^T y$, e uma direção factível de descida, definida como $\Delta x = -\hat{X}\bar{c}$.

Para encontrar o novo ponto, deve-se calcular o tamanho máximo de passo, dado pela primeira componente do vetor $\begin{bmatrix} x \\ s \end{bmatrix}$ a se anular, isto é:

$$\alpha = Min_j\{ \; Max[\; -x_j/\Delta x_j, \; s_j/\Delta x_j \;] \; \}$$

Tendo em vista os conceitos acima, o algoritmo afim primal canalizado pode ser resumido na seqüência de passos a seguir.

Dado (x, s) um ponto inicial interior factível para P_c, $\beta = 0,995$ e $\epsilon = 10^{-8}$

$k \leftarrow 0$

Repita

$s \longleftarrow d - x$

$\hat{X} \longleftarrow diag(\hat{x}_j)$

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

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

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

$\alpha \longleftarrow \beta \ Min_j \{Max[-x_j/\Delta x_j, s_j/\Delta x_j]\}$

$x \longleftarrow x + \alpha \ \Delta x$

$k \longleftarrow k + 1$

Até $(|x^T\bar{c}| < \epsilon)$

onde

$$\hat{x}_j = \frac{x_j^2 s_j^2}{x_j^2 + s_j^2} \quad j = 1, ..., n$$

2.5.2 Método Afim Dual

Seja o seguinte problema dual equivalente D'_c,

$$(D'_c)\begin{cases} Maximizar & b^T y - d^T w \\ Sujeito\ a & \begin{bmatrix} A^T & -I & I \end{bmatrix} \begin{bmatrix} y \\ w \\ z \end{bmatrix} = c \\ & w, z \geq 0 \end{cases}$$

onde z é o vetor de variáveis de folga associadas às restrições do dual.

Considerando o problema dual equivalente D'_c, o **Teorema das Folgas Complementares** (Luenberger [64]) estabelece que, se existirem os pontos (x^*, s^*) e (y^*, w^*, z^*),

soluções factíveis de P_c e D'_c, satisfazendo a condição

$$\begin{aligned} Z^*x^* &= 0 \quad \text{onde} \quad Z^* = diag(z^*) \\ W^*s^* &= 0 \quad \text{onde} \quad W^* = diag(w^*), \end{aligned}$$

então (x^*, s^*) é solução ótima do primal e (y^*, w^*, z^*) é solução ótima do dual.

Seja (y, w, z) um ponto interior dual factível (isto é, y irrestrito, $w, z > 0$ e $A^T y - w + z = c$) e $h(x, s)$ o vetor dos desvios da condição de folgas complementares. Ou seja,

$$h(x,s) = \begin{bmatrix} Zx \\ Ws \end{bmatrix}$$

onde $Z = diag(z)$, $z = c - A^T y + w$ e $W = diag(w)$.

Na tentativa de atender a condição de folgas complementares, pode-se obter o vetor de variáveis primais $(\hat{x}, \hat{s})$ que minimizem o quadrado do desvio $h(x, s)$, restrito a $Ax = b$ e $x + s = d$. Para isso, resolve-se o problema quadrático restrito a seguir.

$$\begin{aligned} Minimizar \quad & H(x,s) = h^T(x,s)h(x,s) \\ Sujeito\ a \quad & Ax = b \\ & x + s = d \end{aligned} \tag{2.59}$$

onde $H : R^{n+n} \longrightarrow R$ é uma função quadrática, convexa e não negativa. A solução de (2.59) pode ser obtida pelas condições de estacionariedade do Lagrangeano $L(x, s, \lambda_1, \lambda_2)$,

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

As condições de otimalidade para (2.59) (Luenberger [64]), são:

$$\begin{array}{lcl} \nabla_x L(x,s,\lambda_1,\lambda_2) = 0 & \Rightarrow & \nabla_x H(x,s) + A^T\lambda_1 + \lambda_2 = 0 \\ \nabla_s L(x,s,\lambda_1,\lambda_2) = 0 & \Rightarrow & \nabla_s H(x,s) + \lambda_2 = 0 \\ \nabla_{\lambda_1} L(x,s,\lambda_1,\lambda_2) = 0 & \Rightarrow & Ax - b = 0 \\ \nabla_{\lambda_2} L(x,s,\lambda_1,\lambda_2) = 0 & \Rightarrow & x + s - d = 0 \end{array}$$

Ou seja,

$$2Z^2x + A^T\lambda_1 + \lambda_2 = 0 \tag{2.60}$$

$$2W^2s + \lambda_2 = 0 \tag{2.61}$$

$$Ax - b = 0 \tag{2.62}$$

$$x + s - d = 0 \tag{2.63}$$

De (2.61),

$$\lambda_2 = -2W^2s \tag{2.64}$$

Substituindo a expressão (2.64) em (2.60), temos:

$$2Z^2x + A^T\lambda_1 - 2W^2s = 0$$

ou

$$Z^2x - W^2s + \tfrac{1}{2}A^T\lambda_1 = 0 \tag{2.65}$$

Pré-multiplicando a relação (2.63) por W^2, obtemos:

$$W^2x + W^2s - W^2d = 0 \tag{2.66}$$

Somando (2.65) e (2.66),

$$(Z^2 + W^2)x + \tfrac{1}{2}A^T\lambda_1 = W^2d$$

então

$$\hat{x} = (Z^2 + W^2)^{-1}\ [W^2d - \tfrac{1}{2}A^T\lambda_1]$$

Substituindo $\hat{x}$ na relação (2.62), obtemos:

$$A(Z^2 + W^2)^{-1}[W^2d - \frac{1}{2}A^T\lambda_1] = b$$

Portanto,

$$\lambda_1 = -2\ [A(Z^2 + W^2)^{-1}A^T]^{-1}\ [\ b - A(Z^2 + W^2)^{-1}W^2d\ ]$$

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

$$\begin{aligned} \lambda_1 &= -2(A\hat{Y}A^T)^{-1}[b - A\hat{Y}W^2d] \\ \hat{x} &= \hat{Y}\{\ W^2d + A^T(A\hat{Y}A^T)^{-1}(b - \hat{Y}W^2d)\ \} \end{aligned}$$

onde $\hat{Y} = (Z^2 + W^2)^{-1} = diag(\frac{1}{z_j^2+w_j^2}),\ 1 \leq j \leq n$

A partir de um ponto (y, w), dual factível, pode-se calcular:

- as variáveis de folga, $z = c - A^Ty + w \qquad [\ \bar{c} = c - A^Ty = z - w\ ]$,
- as direções de movimento, Δy, Δw, Δz,

$$\begin{aligned} \Delta y &= (A\hat{Y}A^T)^{-1}\ [b\ -\ A\hat{Y}W^2d\ ] \\ \Delta w &= -W^2(d - \hat{x}),\ \ onde\ \ \hat{x} = \hat{Y}(W^2d + A^T\Delta y) \\ \Delta z &= -Z^2\hat{x}, \end{aligned}$$

- e as variáveis primais, $\hat{x} = \hat{Y}(W^2d + A^T\Delta y)$

Tendo em vista os conceitos acima, o algoritmo afim dual canalizado pode ser resumido na seqüência de passos a seguir.

Dado (y, w, z) um ponto inicial interior factível para D'_c, $\beta = 0,995$ e $\epsilon = 10^{-8}$.

$k \leftarrow 0$

Repita

$z \longleftarrow c - A^Ty + w$

$\hat{Y} \longleftarrow diag(1/(w_j + z_j))$

$\Delta y \longleftarrow (A\hat{Y}A^T)^{-1}[b - A\hat{Y}W^2d]$

$x \longleftarrow \hat{Y}(W^2d + A^T\Delta y)$

$\Delta w \longleftarrow -W^2(d - x) \qquad [W = diag(w)]$

$\Delta z \longleftarrow -Z^2x \qquad [Z = diag(z)]$

$\alpha \longleftarrow \beta\ Min[min(-z_j/\Delta z_j : \Delta z_j < 0), min(-w_j/\Delta w_j : \Delta w_j < 0)]$

$y \longleftarrow y + \alpha\ \Delta y$

$w \longleftarrow w + \alpha\ \Delta w$

$k \longleftarrow k + 1$

Até $(|x^Tz| < \epsilon)$

2.5.3 Método Primal Dual

Nesta seção nós estamos interessados em resolver os problemas lineares (P_c) e (D'_c), simultaneamente (Lustig e outros [67]). Considere o problema de programação linear P_c,

$$(P_c)\begin{cases} Minimizar & c^T x \\ Sujeito\ a & \begin{bmatrix} A & 0 \\ I & I \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix} \\ & x, s \geq 0 \end{cases}$$

e seu dual, D'_c,

$$(D'_c)\begin{cases} Maximizar & b^T y - d^T w \\ Sujeito\ a & \begin{bmatrix} A^T & -I & I \end{bmatrix} \begin{bmatrix} y \\ w \\ z \end{bmatrix} = c \\ & w, z \geq 0 \end{cases}$$

O método Primal-Dual pode ser obtido aplicando-se o método barreira logarítmico ao problema primal (P_c), isto é, resolvendo-se a seguinte família de problemas:

$$(P_\mu^c)\begin{cases} Minimizar & c^T x - \mu \sum_j \ln x_j - \mu \sum_j \ln s_j \\ Sujeito\ a & Ax = b \\ & x + s = d \end{cases}$$

onde $\mu > 0$ é um parâmetro de penalização (barreira).

Associando-se as variáveis duais (y, w) às restrições de igualdade, o Lagrangeano associado ao problema P_μ^c, é dado por:

$$L_\mu(x, s, y, w) = c^T x - \mu \sum_j \ln x_j - \mu \sum_j \ln s_j - y^T(Ax - b) - w^T(d - x - s)$$

Aplicando as condições de Kuhn Tucker, para μ fixo, temos:

$$\begin{array}{lllllllll}
\nabla_x L(x,s,y,z) = 0 & \Rightarrow & c & - & \mu X^{-1}e - A^T y & + & w & = & 0 \\
\nabla_s L(x,s,y,z) = 0 & \Rightarrow & & - & \mu S^{-1}e & + & w & = & 0 \\
\nabla_y L(x,s,y,z) = 0 & \Rightarrow & & & Ax & - & b & = & 0 \\
\nabla_z L(x,s,y,z) = 0 & \Rightarrow & & & x+s & - & d & = & 0
\end{array} \tag{2.67}$$

Tendo em vista que $A^T y - w + z = c$, a primeira equação de (2.67) pode ser escrita assim:

$$z - \mu X^{-1}e = 0,$$

onde,

$$z = c - A^T y + w$$

Em resumo, temos:

$$\begin{array}{lllllll}
Ax & & & & & = & b \\
x & + & s & & & = & d \\
A^T y & - & w & + & z & = & c \\
 & & & & XZe & = & \mu e \\
 & & & & SWe & = & \mu e
\end{array}$$

onde $X = diag(x)$, $S = diag(s)$, $W = diag(w)$ e $Z = diag(z)$, são matrizes diagonais.

Supõe-se que uma estimativa das variáveis primais (x, s) e duais (y, w, z) está disponível, satisfazendo ambos factibilidade primal e dual, isto é, $Ax = b$, $x + s = d$, $A^T y - w + z = c$ e $(x, s, w, z) > 0$. Fixando-se μ e aplicando-se um passo do método de Newton ao sistema acima, obtem-se o seguinte sistema de equações:

$$\begin{pmatrix} A & 0 & 0 & 0 & 0 \\ I & I & 0 & 0 & 0 \\ 0 & 0 & A^T & -I & I \\ Z & 0 & 0 & 0 & X \\ 0 & W & 0 & S & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta w \\ \Delta z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \mu e - XZe \\ \mu e - SWe \end{pmatrix}$$

Neste sistema temos $4n + m$ equações e $4n + m$ variáveis.

Fazendo-se manipulações algébricas, obtemos o sistema de ordem m a seguir,

$$(A\theta A^T)\Delta y = A\theta\rho(\mu)$$

onde $\theta = (X^{-1}Z + S^{-1}W)^{-1}$ e $\rho(\mu) = \mu(S^{-1} - X^{-1})e + (Z - W)e$

Além disto,

$$\begin{aligned}
\Delta x &= \theta(A^T\Delta y - \rho(\mu)) \\
\Delta s &= -\Delta x \\
\Delta z &= X^{-1}(\mu e - XZe - Z\Delta x) \\
\Delta w &= S^{-1}(\mu e - SWe + W\Delta x)
\end{aligned}$$

Conhecendo-se Δx, Δs, Δy, Δw, Δz, determina-se uma nova aproximação para as variáveis primais e duais. Assim,

$$\begin{aligned}
x &\longleftarrow x + \alpha_P \Delta x \\
s &\longleftarrow s + \alpha_P \Delta s \\
y &\longleftarrow y + \alpha_D \Delta y \\
w &\longleftarrow w + \alpha_D \Delta w \\
z &\longleftarrow z + \alpha_D \Delta z
\end{aligned}$$

onde α_P e α_D são os tamanhos de passos nos espaços primal e dual, respectivamente, escolhidos de tal forma que as variáveis (x, s, w, z) sejam positivas. Lustig e outros [70] sugerem que a cada passo o parâmetro μ seja reduzido de acordo com o critério a seguir:

$$\mu = \frac{c^T x - b^T y + d^T w}{n^2}$$

O numerador da equação acima é normalmente definido como *gap* de dualidade. O parâmetro no denominador, n^2, é o número de colunas da matriz A (elevado ao quadrado).

Tendo em vista os conceitos acima, o algoritmo primal–dual canalizado pode ser resumido na seqüência de passos a seguir.

Dado (x, s, y, w, z) um ponto interior primal-dual factível, $\beta = 0,9995$ e $\epsilon = 10^{-8}$

$k \longleftarrow 0$

Repita

$\mu \longleftarrow (c^T x - b^T y + d^T w)/n^2$

$\theta \longleftarrow (X^{-1}Z + S^{-1}W)^{-1}$

$\Delta y \longleftarrow (A\theta A^T)^{-1} A\theta\rho(\mu), \qquad \rho(\mu) = \mu(S^{-1} - X^{-1})e + (Z - W)e$

$\Delta x \longleftarrow \theta(A^T \Delta y - \rho(\mu))$

$\Delta z \longleftarrow X^{-1}(\mu e - XZe - Z\Delta x)$

$\Delta w \longleftarrow S^{-1}(\mu e - SWe + W\Delta x)$

$\Delta s \longleftarrow -\Delta x$

$\alpha_P \longleftarrow \beta \; Min[max(-x_j/\Delta x_j, s_j/\Delta x_j)]$

$\alpha_D \longleftarrow \beta \; Min[min(-z_j/\Delta z_j : \Delta z_j < 0), min(-w_j/\Delta w_j : \Delta w_j < 0)]$

$x \longleftarrow x + \alpha_P \; \Delta x$

$s \longleftarrow s + \alpha_P \; \Delta s$

$y \longleftarrow y + \alpha_D \; \Delta y$

$w \longleftarrow w + \alpha_D \; \Delta w$

$z \longleftarrow z + \alpha_D \; \Delta z$

$k \longleftarrow k + 1$

Até $(|x^T z| < \epsilon)$

Se não tivessemos factibilidade primal e dual, isto é, se não existisse (x, s, y, w, z) com $(x, s, w, z) > 0$ tais que $x + s = d$, então o sistema de equações a ser resolvido seria:

$$\begin{pmatrix} A & 0 & 0 & 0 & 0 \\ I & I & 0 & 0 & 0 \\ 0 & 0 & A^T & -I & I \\ Z & 0 & 0 & 0 & X \\ 0 & W & 0 & S & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta w \\ \Delta z \end{pmatrix} = \begin{pmatrix} b - Ax \\ 0 \\ c - A^T y - z + w \\ \mu e - XZe \\ \mu e - SWe \end{pmatrix}$$

Novamente, obtemos um sistema com $4n + m$ equações e $4n + m$ variáveis. Fazendo-se manipulações algébricas, obtemos o seguinte sistema de ordem m:

$$(A\theta A^T)\Delta y = A\theta(\rho(\mu) + d_D) + d_P$$

onde $\theta = (X^{-1}Z + S^{-1}W)^{-1}$ e $\rho(\mu) = \mu(S^{-1} - X^{-1})e + (Z - W)e$

Além disto,

$$\begin{array}{lcl} \Delta x & = & \theta \ (\ A^T\Delta y - \rho(\mu) - d_D \ ) \\ \Delta z & = & X^{-1} \ (\ \mu e - XZe - Z\Delta x \ ) \\ \Delta w & = & S^{-1} \ (\ \mu e - SWe + W\Delta x \ ) \\ \Delta s & = & -\Delta x \end{array}$$

O cálculo dessas direções caracteriza uma outra versão do algoritmo primal-dual, onde apenas $x + s = d$, precisa ser satisfeita desde o início do algoritmo (Lustig e outros [70]).

Esta implementação do algoritmo primal-dual canalizado, adotada neste trabalho, pode ser resumida na seqüência de passos a seguir.

Dado um ponto interior (x, s, w, z) (isto é $(x, s, w, z) > 0$), tal que $x + s = d$, $\beta = 0,9995$ e $\epsilon = 10^{-8}$,

$k \leftarrow 0$

Repita

$\mu \longleftarrow (c^Tx - b^Ty + d^Tw)/n^2$

$\theta \longleftarrow (X^{-1}Z + S^{-1}W)^{-1}$

$\Delta y \longleftarrow (A\theta A^T)^{-1}(A\theta(\rho(\mu) + d_D) + d_P)$

$\Delta x \longleftarrow \theta(A^T\Delta y - \rho(\mu) - d_D)$

$\Delta z \longleftarrow X^{-1}(\mu e - XZe - Z\Delta x)$

$\Delta w \longleftarrow S^{-1}(\mu e - SWe + W\Delta x)$

$\Delta s \longleftarrow -\Delta x$

$\alpha_P \longleftarrow \beta \ Min[max(-x_j/\Delta x_j, s_j/\Delta x_j)]$

$\alpha_D \longleftarrow \beta \ Min[min(-z_j/\Delta z_j : \Delta z_j < 0), min(-w_j/\Delta w_j : \Delta w_j < 0)]$

$x \longleftarrow x + \alpha_P \ \Delta x$

$s \longleftarrow s + \alpha_P \ \Delta s$

$y \longleftarrow y + \alpha_D \ \Delta y$

$w \longleftarrow w + \alpha_D \ \Delta w$

$z \longleftarrow z + \alpha_D \ \Delta z$

$k \longleftarrow k + 1$

Até $(|x^Tz| < \epsilon)$

onde $d_P = b - Ax$, $d_D = c - A^T y - z + w$, $\rho(\mu) = \mu(S^{-1} - X^{-1})e + (Z - W)e$

2.5.4 Método Preditor-Corretor

Considere novamente o par de problemas (P_c, D'_c). No método primal-dual preditor-corretor não é necessário que as restrições de canalização sejam satisfeitas desde o início. Neste caso, a nova solução (x, s, y, w, z), obtida a partir das direções Δx, Δs, Δy, Δw, Δz,

$$\begin{aligned} x &\longleftarrow x + \Delta x \\ s &\longleftarrow s + \Delta s \\ y &\longleftarrow y + \Delta y \\ w &\longleftarrow w + \Delta w \\ z &\longleftarrow z + \Delta z \end{aligned}$$

deve satisfazer as relações a seguir.

$$\begin{aligned} Ax &= b \\ x + s &= d \\ A^T y - w + z &= c \\ XZe &= \mu e \\ SWe &= \mu e \end{aligned}$$

Substituindo-se a nova solução $(x + \Delta x, s + \Delta s, y + \Delta y, w + \Delta w, z + \Delta z)$, nas equações acima, temos:

$$\begin{aligned} A(x + \Delta x) &= b \\ (x + \Delta x) + (s + \Delta s) &= d \\ A^T(y + \Delta y) - (w + \Delta w) + (z + \Delta z) &= c \\ (X + \Delta X)(Z + \Delta Z)e &= \mu e \\ (S + \Delta S)(W + \Delta W)e &= \mu e \end{aligned} \tag{2.68}$$

onde $\Delta X = diag(\Delta x)$, $\Delta Z = diag(\Delta z)$, $\Delta S = diag(\Delta s)$ e $\Delta W = diag(\Delta w)$ são matrizes diagonais.

As equações (2.68) são equivalentes ao seguinte sistema matricial:

$$\begin{pmatrix} A & 0 & 0 & 0 & 0 \\ I & I & 0 & 0 & 0 \\ 0 & 0 & A^T & -I & I \\ Z & 0 & 0 & 0 & X \\ 0 & W & 0 & S & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta w \\ \Delta z \end{pmatrix} = \begin{pmatrix} b - Ax \\ d - x - s \\ c - A^T y - z + w \\ \mu e - XZe - \Delta X \Delta Ze \\ \mu e - SWe - \Delta S \Delta We \end{pmatrix} \tag{2.69}$$

O sistema acima tem $4n + m$ equações e $4n + m$ variáveis. No lado direito aparecem os termos não lineares $(\Delta X \Delta Ze, \Delta S \Delta We)$. Esses termos não são conhecidos, no entanto, podemos calcular uma aproximação para os mesmos, usando o método primal-dual afim escala ($\mu = 0$). Isto corresponde à etapa de predição do método preditor-corretor proposto por Mehrotra [77], já discutido. Assim, o sistema de equações a ser resolvido na etapa de predição é dado por,

$$\begin{pmatrix} A & 0 & 0 & 0 & 0 \\ I & I & 0 & 0 & 0 \\ 0 & 0 & A^T & -I & I \\ Z & 0 & 0 & 0 & X \\ 0 & W & 0 & S & 0 \end{pmatrix} \begin{pmatrix} \delta x \\ \delta s \\ \delta y \\ \delta w \\ \delta z \end{pmatrix} = \begin{pmatrix} d_P \\ d_C \\ d_D \\ -XZe \\ -SWe \end{pmatrix}$$

onde

$$\begin{aligned} d_P &= b - Ax \\ d_C &= d - x - s \\ d_D &= c - A^T y + w - z \end{aligned}$$

e $(\delta x, \delta s, \delta y, \delta w, \delta z)$ é uma aproximação das direções de deslocamento $(\Delta x, \Delta s, \Delta y, \Delta w, \Delta z)$.

Fazendo manipulações algébricas, obtemos o seguinte sistema de equações de ordem m:

$$(A\theta A^T)\delta y = A\theta[(Z - W)e + d_D - S^{-1}Wd_C) + d_P$$

onde $\theta = (X^{-1}Z + S^{-1}W)^{-1}$

Além disto,

$$\begin{aligned} \delta x &= \theta(A^T \delta y - (Z - W)e - d_D + S^{-1}Wd_C)z \\ \delta s &= d_C - \delta x \\ \delta z &= X^{-1}(-XZe - Z\delta x) \\ \delta w &= S^{-1}(-SWe - W\delta s) \end{aligned}$$

Uma vez obtida uma aproximação dos termos não lineares $\Delta X \Delta Z e$ e $\Delta S \Delta W e$, denotado por $\delta X \delta Z e$ e $\delta S \delta W e$, respectivamente, deve-se substituir a aproximação no lado direito da equação matricial (2.69) e resolver o sistema. Esta é a etapa de correção. O novo sistema a ser resolvido é,

$$\begin{pmatrix} A & 0 & 0 & 0 & 0 \\ I & I & 0 & 0 & 0 \\ 0 & 0 & A^T & -I & I \\ Z & 0 & 0 & 0 & X \\ 0 & W & 0 & S & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta w \\ \Delta z \end{pmatrix} = \begin{pmatrix} d_P \\ d_C \\ d_D \\ \mu e - XZe - \delta X \delta Z e \\ \mu e - SWe - \delta S \delta W e \end{pmatrix} \tag{2.70}$$

onde $\delta X = diag(\delta x), \delta Z = diag(\delta z), \delta S = diag(\delta s), \delta X = diag(\delta x)$, são matrizes diagonais.

Fazendo-se manipulações algébricas, obtemos:

$$\Delta y = (A\theta A^T)^{-1}(A\theta[\rho(\mu) + d_D - S^{-1}W d_C + X^{-1}\delta X \delta Z - S^{-1}\delta S \delta W] + d_P)$$

onde $\rho(\mu) = \mu(S^{-1} - X^{-1})e + (Z - W)e$

Além disto,

$$\begin{aligned} \Delta x &= \theta(A^T \Delta y - \rho(\mu) - d_D + S^{-1}W d_C - X^{-1}\delta X \delta Z e + S^{-1}\delta S \delta W e) \\ \Delta s &= d_C - \Delta x \\ \Delta z &= X^{-1}(\mu e - XZe - \delta X \delta Z e - Z\Delta x) \\ \Delta w &= S^{-1}(\mu e - SWe - \delta S \delta W e - W \Delta s) \end{aligned}$$

Em resumo, temos:

- **Predição** onde se calcula uma aproximação dos termos não lineares $(\Delta X \Delta Z e, \Delta S \Delta W e)$, denotado por $(\delta X \delta Z e, \delta S \delta W e)$ usando o método primal-dual afim escala, isto equivale a aplicar o algoritmo primal-dual, apresentado no item 2.5.3, com o parâmetro de penalização $\mu = 0$.
- **Correção** usando-se a aproximação obtida na etapa de predição, deve-se resolver o sistema de equações (2.70), para obter as direções de deslocamento primais duais $(\Delta x, \Delta s, \Delta y, \Delta w, \Delta z)$.
- **Tamanho de passo** deve-se calcular α_P e α_D, os tamanhos de passos no espaços primal e dual, respectivamente, escolhidos de tal forma que o novo ponto (x, s, y, w, z) seja ainda interior.

- **Novo ponto** finalmente deve-se calcular o novo ponto.

O algoritmo primal-dual preditor-corretor com variáveis canalizadas pode ser resumido na seqüência de passos a seguir.

Dado um ponto interior (x, s, w, z) (isto é $(x, s, w, z) > 0$), $\beta = 0,9995$ e $\epsilon = 10^{-8}$

$k \leftarrow 0$

Repita

$$\mu \longleftarrow (c^T x - b^T y + d^T w)/n^2$$
$$\theta \longleftarrow (X^{-1}Z + S^{-1}W)^{-1}$$
$$\delta y \longleftarrow (A\theta A^T)^{-1}(A\theta[(Z - W)e + d_D - S^{-1}W d_C] + d_P)$$
$$\delta x \longleftarrow \theta(A^T \delta y - (Z - W)e - d_D + S^{-1}W d_C)$$
$$\delta s \longleftarrow d_C - \delta x$$
$$\delta z \longleftarrow X^{-1}(-XZe - Z\delta x)$$
$$\delta w \longleftarrow S^{-1}(-SWe - W\Delta s)$$
$$\Delta y \longleftarrow (A\theta A^T)^{-1}(A\theta(\rho(\mu) + d_D - S^{-1}W d_C + X^{-1}\delta X \delta Ze - S^{-1}\delta S \delta We) + d_P)$$
$$\Delta x \longleftarrow \theta(A^T \Delta y - \rho(\mu) - d_D + S^{-1}W d_C - X^{-1}\delta X \delta Ze + S^{-1}\delta S \delta We)$$
$$\Delta s \longleftarrow d_C - \Delta x$$
$$\Delta z \longleftarrow X^{-1}(\mu e - XZe - \delta X \delta Ze - Z\Delta x)$$
$$\Delta w \longleftarrow S^{-1}(\mu e - SWe - \delta S \delta We - W\Delta s)$$
$$\alpha_P \longleftarrow \beta \; Min[max(-x_j/\Delta x_j, s_j/\Delta x_j)]$$
$$\alpha_D \longleftarrow \beta \; Min[min(-z_j/\Delta z_j : \Delta z_j < 0), min(-w_j/\Delta w_j : \Delta w_j < 0)]$$
$$x \longleftarrow x + \alpha_P \; \Delta x$$
$$s \longleftarrow s + \alpha_P \; \Delta s$$
$$y \longleftarrow y + \alpha_D \; \Delta y$$
$$w \longleftarrow w + \alpha_D \; \Delta w$$
$$z \longleftarrow z + \alpha_D \; \Delta z$$
$$k \longleftarrow k + 1$$

Até $(|x^T z| < \epsilon)$

O algoritmo pode iniciar o processo com uma solução inicial interior primal infactível e dual infactível (isto é, satisfazendo apenas $(x, s, w, z) > 0$). Mas, neste caso, é possível achar

uma solução inicial interior dual factível de forma muito simples. Seja,

$$y = (y_i),\ c = (c_j),\ w = (w_j) \text{ e } z = (z_j)$$

Tem-se a seguinte solução interior dual factível:

$$\begin{array}{llll} y_i = 0 & \forall i & & \\ \texttt{Se} & c_j > 0, & \texttt{então} & \{w_j = 1 \quad e \quad z_j = 1 + c_j\} \\ \texttt{Caso} & \texttt{contrário}, & & \{z_j = 1 \quad e \quad w_j = 1 - c_j\} \end{array}$$

2.6 Soluções Iniciais

Os métodos afins (afim primal e afim dual) precisam de uma solução inicial interior factível, para iniciar o processo de solução dos problemas P e D', respectivamente.

No método afim primal deve-se fazer um procedimento semelhante à fase 1 do método simplex. Ou seja, modifica-se o problema de tal forma que seja possível aplicar o próprio método ao problema modificado, para encontrar um ponto interior factível do problema original. Assim, denominaremos o processo de inicialização de fase 1.

Suponha que $x^0 > 0$ qualquer. Para se obter uma solução inicial factível pode-se resolver o seguinte problema artificial, P_a,

$$(P_a) \begin{cases} Minimizar & \lambda & & \\ Sujeito\ a & Ax + \lambda\rho & = & b \\ & x, \lambda & \geq & 0 \end{cases}$$

onde $\rho = b - Ax^0$.

No método afim dual, devemos fazer um procedimento análogo à fase 1 do método simplex, neste caso, resolve-se o seguinte problema artificial, D_a,

$$(D_a) \begin{cases} Minimizar & \lambda & & \\ Sujeito\ a & A^T y + z + \lambda\rho & = & c \\ & z, \lambda & \geq & 0 \end{cases}$$

onde $\rho = c - A^T y^0 - z^0$, e (y^0, z^0) tal que $z^0 > 0$.

No método primal-dual simples não é necessário ter um ponto primal-dual factível. É possível começar o processo iterativo com um ponto interior (x, y, z) tal $x > 0$ e $z > 0$. Ou seja, com um ponto primal-dual interior infactível. As direções de deslocamento Δy, Δz, Δx levam em conta as infactibilidades do primal e do dual.

Assim, a direção de deslocamento Δy é obtida resolvendo o seguinte sistema:

$$(AZ^{-1}XA^T)\Delta y = b + AZ^{-1}Xd_D - \mu AZ^{-1}e$$

onde $d_D = c - A^T y - z$ é a infactibilidade do dual.

Finalmente, no método preditor corretor não é preciso ter uma solução primal dual factível; as direções de deslocamento levam em conta as infactibilidades primal e dual.

2.7 Comentários

Todos os métodos apresentados nas seções anteriores, levam à solução de um sistema simétrico e positivo definido do tipo,

$$AD^*A^T y = \hat{b} \tag{2.71}$$

onde D^* é uma matriz diagonal (e positiva definida), o vetor Δy é a direção de deslocamento (solução do sistema) e o vetor $\hat{b}$ é o lado direito do sistema.

No método afim primal,

$$D^* = X^2 \quad e \quad \hat{b} = AX^2c$$

No método afim dual,

$$D^* = Z^{-2} \quad e \quad \hat{b} = b$$

No método primal-dual afim escala,

$$D^* = Z^{-1}X \quad e \quad \hat{b} = b + AZ^{-1}d_D$$

No método afim primal canalizado,

$$D^* = \hat{X} \quad e \quad \hat{b} = A\hat{X}c$$

No método afim dual canalizado,

$$D^* = \hat{Y} \quad e \quad \hat{b} = b - A\hat{Y}W^2d$$

No método primal dual canalizado,

$$D^* = \theta \quad e \quad \hat{b} = A\theta(\rho(\mu) + d_D) + d_P)$$

No método primal-dual preditor corretor deve-se resolver dois sistemas do tipo (2.71). O primeiro sistema a ser resolvido (etapa de predição) é dado por:

$$(A\theta A^T)\delta y = AXe + AZ^{-1}Xd_D + d_P$$

O segundo sistema (etapa de correção) é dado por:

$$(A\theta A^T)\Delta y \quad = \quad b + AZ^{-1}Xd_D + AZ^{-1}X\delta X\delta Ze - \mu Z^{-1}Xe$$

Ou seja, no método primal-dual preditor-corretor, a cada iteração deve-se resolver dois sistemas simétricos e positivo definidos da forma (2.71) com a mesma matriz de coeficientes, porém com lados direitos ($\hat{b}$) distintos.

O estudo da solução do sistema (2.71) é objeto do próximo capítulo.

Capítulo 3

Solução de $(AD^*A^T)y = \hat{b}$ em Redes

3.1 Introdução

O objetivo deste capítulo é estudar os métodos para solução de sistemas do tipo $(AD^*A^T)y = \hat{b}$, onde a matriz A é a matriz de incidência associada a um grafo conexo com m nós e n arcos orientados (vide apêndice A). É importante salientar que a matriz de incidência não é de posto completo, entretanto a eliminação de uma linha qualquer da matriz transforma-a numa matriz de posto completo (Kennington e Helgason [58]). Neste trabalho denotamos por A a matriz obtida da matriz de incidência pela eliminação de sua última linha (m-ésima linha).

O principal interesse no estudo destes sistemas é obter melhorias na eficiência dos algoritmos de resolução de problemas de fluxos de custo mínimo através dos métodos de pontos interiores discutidos no capítulo anterior.

A maior parte do esforço computacional dos algoritmos baseados nos métodos de pontos interiores é dedicado ao cálculo de AD^*A^T e à solução de sistemas do tipo $(AD^*A^T)y = \hat{b}$, onde D^* é uma matriz diagonal.

3.2 Idéia Geral dos Métodos de Solução

A resolução do sistema $(AD^*A^T)y = \hat{b}$ pode ser obtida através de métodos diretos ou métodos iterativos. Os métodos diretos são variantes da decomposição $LU = LDL^T = \hat{L}\hat{L}^T$ (fatoração de Choleski), onde L é triangular inferior com elementos da diagonal iguais a um (1), $\hat{L}$ é triangular inferior, D é uma matriz diagonal e U é triangular superior (Duff e outros [32]).

A solução do sistema $AD^*A^Ty = \hat{b}$ é obtida resolvendo o sistema equivalente:

$$LDL^Ty = \hat{b}$$

Para encontrar a solução do sistema acima, faz-se $z = L^Ty$ e $w = Dz$. Inicialmente determina-se w tal que $Lw = \hat{b}$, em seguida faz-se $z = D^{-1}w$ e finalmente resolve-se $L^Ty = z$.

Existe uma outra família de métodos, denominados iterativos (Luenberger [64]), esses métodos são baseados no fato de que a solução do sistema $(AD^*A^T)y = \hat{b}$ pode ser encontrada resolvendo o problema abaixo,

$$Minimizar \;\; F(y) = \frac{1}{2}y^T(AD^*A^T)y - y^T\hat{b},$$

onde AD^*A^T é Simétrica e Positiva Definida.

Observe que a função F é uma função convexa quadrática e, portanto, tem um único ponto de mínimo $y^* = (AD^*A^T)^{-1}\hat{b}$. Assim,

$\nabla F(y^*) = 0$ implica $y^* = (AD^*A^T)^{-1}\hat{b}$ (solução de $AD^*A^Ty = \hat{b}$).

Ambos os métodos, diretos e iterativos, serão analizados neste trabalho.

3.3 Estrutura de $Q = AD^*A^T$

A matriz A é uma matriz de incidência associada ao grafo orientado G, com m nós (aos quais estão associados as linhas de A) e n arcos (aos quais estão associados as colunas de A). As colunas de A têm no máximo dois elementos não nulos, $+1$ na linha

correspondente ao nó origem do arco e -1 na linha correspondente ao nó destino do arco. O grafo G (e consequentemente a matriz A) pode ser armazenado em dois vetores, t e h, de dimensão n. Para cada arco j, $t_j = i$ e $h_j = k$, sendo i o nó origem do arco j ($A_{ij} = +1$) e k o nó destino do arco j ($A_{kj} = -1$). Assim, $Q = AD^*A^T$ é uma matriz quadrada simétrica, de dimensão m com linhas e colunas associadas aos nós do grafo G. Essa matriz pode ser facilmente obtida a partir dos vetores t e h. Neste caso, a matriz Q é simétrica e positiva definida com elementos,

$$Q_{ij} = \begin{cases} -\sum_k [d_k^* : t_k = i, h_k = j] - \sum_k [d_k^* : t_k = j, h_k = i] & i \neq j \\ \sum_k [d_k^* : t_k = i] + \sum_k [d_k^* : h_k = i] & i = j \end{cases}$$

Observe que, no caso particular de $D^* = I$, um elemento da diagonal Q_{ii}, é igual ao grau do nó correspondente (i). Um elemento fora da diagonal, $Q_{ik} = Q_{ki}$, é igual ao número de arcos ligando os nós i e k, com sinal negativo (independente do sentido dos arcos).

Por exemplo, considere a rede com 5 nós e 7 arcos, representada na Figura 3.1.

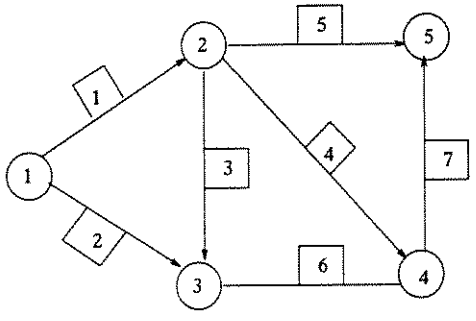


Figura 3.1: Rede associada à matriz de incidência A

Cuja matriz de incidência (incluindo a linha a ser eliminada) é:

$$A = \left(\begin{array}{ccccccc} 1 & 1 & & & & & \\ -1 & & & 1 & 1 & 1 & \\ & -1 & -1 & & & & 1 \\ \hline & & & -1 & & -1 & 1 \\ & & & & -1 & & -1 \end{array}\right)$$

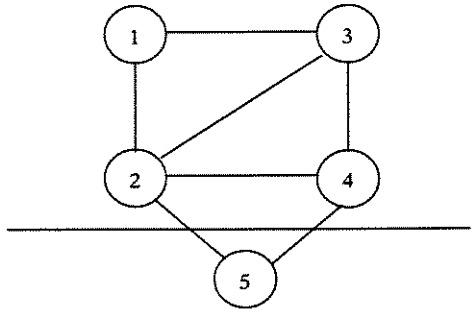
A matriz AIA^T (incluindo a linha e a coluna a serem eliminadas) é:

$$AIA^T = \left(\begin{array}{rrrr|r} 2 & -1 & -1 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ \hline 0 & -1 & 0 & -1 & 2 \end{array}\right)$$

Considere a matriz $Q = AD^*A^T$. Pode-se associar à matriz Q um grafo não orientado H, representando a localização dos elementos não nulos na matriz (Duff e outros [32]). Os nós do grafo H correspondem às linhas (e colunas) da matriz Q e cada arco não orientado $\{i, j\}$ de H corresponde a um par de elementos Q_{ij}, Q_{ji} não nulos de Q. Não são considerados os elementos Q_{ii} da diagonal.

Observe que, H tem um nó correspondente a cada nó do grafo G, e tem um arco não orientado entre os nós i e k sempre que exista pelo menos um arco (orientado) entre i e k em G. Vale enfatizar que, independente da quantidade de arcos entre os nós i e k em G, existe apenas um arco não orientado entre os nós correspondentes em H.

Dando continuidade ao exemplo acima, o grafo associado H (incluindo o nó 5 a ser eliminado) é:



3.4 Solução por Métodos Diretos

Métodos diretos são aqueles que, a menos de erros de arredondamento, fornecem a solução exata do sistema $Qy = \hat{b}$, após um número finito de iterações. Neste trabalho,

discutiremos apenas a decomposição de Choleski (Golub e Van Loan [46], Duff e outros [32]).

3.4.1 Decomposição de Choleski

Seja Q uma matriz simétrica e positiva definida de ordem m. A decomposição de Choleski da matriz Q, consiste em determinar uma matriz triangular inferior L (com elementos positivos na diagonal), tal que $Q = LL^T$. Os elementos da matriz $L = (L_{ij})$ podem ser calculados pela fórmula recursiva:

$$\begin{aligned} L_{jj} &= \sqrt{Q_{jj} - \sum_{k=1}^{j-1} L_{jk}^2} \qquad j = 1, ..., m \\ L_{ij} &= \frac{1}{L_{jj}}(Q_{ij} - \sum_{k=1}^{j-1} L_{ik} * L_{jk}) \quad i > j \end{aligned}$$

Para calcular os elementos da matriz triangular inferior L, pode-se aplicar o seguinte algoritmo (Golub e Van Loan [46], Duff e outros [32]).

$$\left[\begin{array}{l} L_{11} = \sqrt{Q_{11}} \\ \texttt{for}\ \ j = 2\ \texttt{to}\ m\ \texttt{do}\ L_{j1} = Q_{j1}/L_{11} \\ \texttt{for}\ \ j = 2\ \texttt{to}\ m\ \texttt{do} \\ \qquad S = 0 \\ \qquad \texttt{for}\ \ k = 1\ \ \texttt{to}\ \ (j-1)\ \ \texttt{do}\ \ S = S + L_{jk}^2 \\ \qquad L_{jj} = \sqrt{Q_{jj} - S} \\ \qquad \texttt{for}\ \ i = j+1\ \ \texttt{to}\ m\ \texttt{do} \\ \qquad\qquad S = 0 \\ \qquad\qquad \texttt{for}\ \ k = 1\ \ \texttt{to}\ \ (j-1)\ \texttt{do}\ \ S = S + L_{ik} * L_{jk} \\ \qquad\qquad L_{ij} = (Q_{ij} - S)/L_{jj} \end{array}\right.$$

Neste caso, para resolver o sistema $Qy = \hat{b}$. Inicialmente obtem-se a decomposição de Choleski da matriz Q, em seguida faz-se $w = L^T y$ e a seguir resolvem-se os seguintes sistemas triangulares:

$$\begin{aligned} Lw &= \hat{b} \quad (sistema\ triangular\ inferior) \\ L^T y &= w \quad (sistema\ triangular\ superior). \end{aligned}$$

3.4.2 O Método de Choleski e o Problema de Ordenação

Suponha que o sistema a ser resolvido é,

$$Qy = \hat{b} \tag{3.1}$$

onde Q é uma matriz de coeficientes de ordem m, simétrica e positiva definida, e y é o vetor solução ($y \in R^m$), cujas componentes serão calculadas. A aplicação do método de Choleski à matriz Q leva à fatoração triangular,

$$Q = LL^T \tag{3.2}$$

onde L é triangular inferior com elementos positivos na diagonal.

Usando (3.2) em (3.1), temos:

$$LL^T y = \hat{b} \tag{3.3}$$

Fazendo-se $w = L^T y$, a solução y é obtida através da solução dos seguintes sistemas triangulares:

$$Lw = \hat{b} \tag{3.4}$$

$$L^T y = w \tag{3.5}$$

Como um exemplo, considere o sistema:

$$\begin{pmatrix} 4 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.6}$$

Observe que a estrutura da matriz de coeficientes é:

$$\begin{pmatrix} \times & \times & \times & \times \\ \times & \times & & \\ \times & & \times & \\ \times & & & \times \end{pmatrix}$$

O fator de Choleski da matriz de coeficientes de (3.6) é dado por:

$$L = \begin{pmatrix} 2 & & & \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & & \\ -\frac{1}{2} & -\frac{\sqrt{3}}{6} & \frac{\sqrt{6}}{3} & \\ -\frac{1}{2} & -\frac{\sqrt{3}}{6} & -\frac{\sqrt{6}}{6} & \frac{\sqrt{2}}{2} \end{pmatrix}$$

Resolvendo $Lw = \hat{b}$, obtemos:

$$w = \begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{6} \\ \frac{\sqrt{6}}{6} \\ \frac{\sqrt{2}}{2} \end{pmatrix}$$

Resolvendo-se $L^T y = w$, tem-se:

$$y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

O exemplo acima ilustra o aspecto mais importante acerca da aplicação do método de Choleski para uma matriz esparsa Q: a matriz usualmente sofre preenchimento (*fill-in*). Ou seja, a matriz L tem elementos não nulos em posições onde existiam elementos nulos na parte triangular inferior da matriz Q.

A ocorrência de preenchimento pode ser evitada fazendo-se permutações de linhas/colunas no sistema (3.6).

Suponha que é feita uma permutação das linhas da matriz Q, por exemplo a linha 1 associada à primeira equação é permutada com a linha 5 associada à última equação, e uma permutação correspondente nas colunas (a coluna 1 associada à primeira variável y_1 é permutada com a última coluna associada à variável y_5).

Observe que agora temos que resolver o sistema de equações (3.7), equivalente ao sistema original (3.1),

$$(PQP^T)(Py) = P\hat{b} \tag{3.7}$$

onde Q é a matriz de coeficientes, $\hat{b}$ é o vetor de termos independentes, e P é uma matriz de permutação.

O sistema (3.7), pode ser escrito como,

$$\hat{Q}\hat{y} = \bar{b} \tag{3.8}$$

onde $\hat{Q} = PQP^T$, $\hat{y} = Py$ e $\bar{b} = P\hat{b}$.

Usando, como antes, o método de Choleski para resolver o sistema (3.8), fatora-se $\hat{Q}$ em $\hat{L}\hat{L}^T$, obtendo:

$$\hat{L} = \begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ -1 & -1 & -1 & 1 \end{pmatrix} \tag{3.9}$$

Resolvendo-se $\hat{L}\hat{w} = \bar{b}$ e $\hat{L}^T\hat{y} = \hat{w}$, obtêm-se a solução $\hat{y}$, a qual é simplesmente uma permutação de y.

O ponto crucial é que o reordenamento das equações e variáveis (permutação P), fornece um fator triangular $\hat{L}$ tão esparso quanto o triângulo inferior da matriz Q. Embora na prática isto seja raramente possível, pois a ordenação ótima é um problema NP-Completo (Yannakakis [98]), para muitos problemas com matrizes esparsas um reordenamento das linhas/colunas, que produza um número aceitável de *fill-ins* (poucos *fill-ins*), pode produzir uma grande redução no tempo de execução e requisitos de memória para armazenamento das informações necessárias à solução do sistema (assumindo-se que a esparsidade seja bem explorada).

O estudo de heurísticas que levam à obtenção de bons reordenamentos (no sentido de preservar a esparsidade) para os sistemas que surgem na solução de problemas de fluxo de custo mínimo através de métodos de pontos interiores é um dos tópicos a ser estudados neste item, junto com um estudo de esquemas de armazenamento e necessidades de cálculo para a manipulação dos fatores esparsos $\hat{L}$ fornecidos por estes reordenamentos.

O exemplo acima ilustra as características básicas da eliminação esparsa e o efeito de reordenamento. Como mostra o exemplo, alguns ordenamentos podem produzir

uma redução dramática na quantidade de *fill-ins*. A tarefa de encontrar um *bom* ordenamento, conhecido como o *problema de ordenação*, é o ponto central ao estudo de solução de sistemas esparsos.

Ao longo do presente trabalho serão estudados diversos métodos heurísticos para encontrar um bom ordenamento das linhas/colunas (produzindo um número aceitável de *fill-ins*) para solução do sistema $(AD^*A^T)y = \hat{b}$, associado a um problema de fluxos em redes.

3.4.3 Etapas para Solução do Sistema Esparso $(AD^*A^T)y = \hat{b}$

Considere a matriz

$$Q = AD^*A^T$$

Neste trabalho, estamos interessados em aplicações que produzem seqüências de matrizes Q esparsas, isto é, apresentam uma grande proporção de elementos nulos. De modo característico uma rede com 1000 nós e 10000 arcos produz uma matriz Q de 1000×1000 com no máximo 21000 elementos não nulos.

Neste caso, deve-se resolver o sistema $Qy = \hat{b}$, executando os seguintes passos:

1. *Ordenação*: Determine uma matriz de permutação P tal que PQP^T tenha um fator de Choleski esparso L.
2. *Fatoração Simbólica*: Determine a estrutura da matriz L e escolha uma estrutura de dados que explore a esparsidade de L.
3. *Fatoração Numérica*: Substitua os elementos não nulos da matriz Q na estrutura de dados e calcule a matriz L.
4. *Solução Triangular*: Usando o L calculado, resolva os sistemas triangulares $Lu = P\hat{b}$, $L^Tv = u$ e faça $y = P^Tv$.

É importante frisar que os métodos de pontos interiores necessitam resolver sistemas do tipo $Qy = \hat{b}$ a cada iteração. Entretanto, a estrutura das matrizes Q de cada iteração é a mesma; isto é, o grafo H é o mesmo para todas as iterações. Isto permite

que os passos de ordenação e fatoração simbólica sejam efetuados uma única vez, enquanto que os passos de fatoração numérica e solução triangular são efetuados a cada iteração dos métodos de pontos interiores.

3.4.4 Heurísticas de Ordenação

Definições Preliminares

As definições abaixo são adaptações de conceitos adotados por Tinney e Walker [92] e Ogbuobiri, Tinney e Walker [83].

- **Definição** 1

  g_i – Grau do nó i, é o número de arcos incidentes no nó i.

- **Definição** 2

  v_i – Valência do nó i, é o número de novos arcos (*fill-ins*) que seriam criados em H entre os nós remanescentes se este nó fosse eliminado.

É importante observar que se o número total de *fill-ins* de uma ordenação for nula, ela é ótima. No entanto as ordenações ótimas normalmente levam a preenchimentos da matriz.

Neste item serão apresentados os principais métodos heurísticos encontrados na literatura para reordenamento das linhas/colunas da matriz esparsa Q (simétrica e positiva definida), os quais fornecem um número aceitável de *fill-ins*.

Fazer um pivoteamento na matriz Q equivale a remover um nó do grafo H e criar novos arcos para formar um clique (sub-grafo completo), entre os nós adjacentes.

Considere a estrutura da matriz Q,

$$\begin{pmatrix} \times & \times & \times & & \\ \times & \times & & \times & \times \\ \times & & \times & & \times \\ & \times & & \times & \times \\ & \times & \times & \times & \times \end{pmatrix}$$

cujo grafo associado é ilustrado na Figura 3.2.

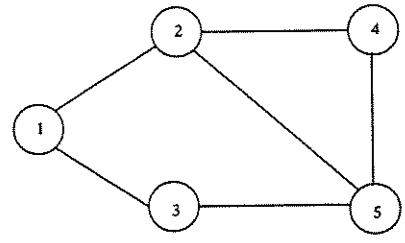


Figura 3.2: Grafo associado à estrutura da matriz Q

Suponha que é feito um pivoteamento no elemento $(1,1)$ da matriz Q. A estrutura da matriz Q após o primeiro pivoteamento é:

$$\begin{pmatrix} \otimes & \times & \times & & \\ & \times & \bullet & \times & \times \\ & \bullet & \times & & \times \\ & \times & & \times & \times \\ & \times & \times & \times & \times \end{pmatrix}$$

O grafo associado à matriz Q após o primeiro pivoteamento é ilustrado na Figura 3.3.

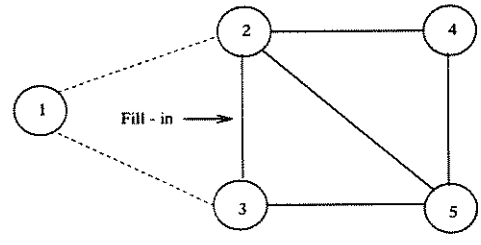


Figura 3.3: Estrutura após o primeiro pivoteamento

Tinney 1 (Esquema 1)

A idéia do processo de reordenamento denominado *Tinney 1* é eliminar os nós do grafo H na seqüência crescente dos graus originais. Se dois ou mais nós apresentam o

mesmo grau, a escolha entre eles é arbitrária (Ogbuobiri, Tinney e Walker [83]).

O enunciado do esquema *Tinney 1* em termos matriciais seria:

Numere as linhas de acordo com o número de elementos não nulos fora da diagonal de cada linha. O processo de eliminação é efetuado na seqüência crescente das linhas renumeradas.

Considere a seqüência de matrizes reduzidas $Q_1, Q_2, Q_3, ..., Q_{m-1}$, obtidas através dos $m-1$ pivoteamentos diagonais (a escolha do pivô é na diagonal) efetuados para obter o fator L da decomposição LU de Q. Considere também a seqüência de grafos residuais $H_1, ..., H_{m-1}$ associados às matrizes $Q_1, Q_2, Q_3, ..., Q_{m-1}$ (respectivamente). Isto significa que H_k é obtido a partir de H_{k-1} eliminando um nó e acresentando arcos correspondentes aos *fill-ins* criados. Note que tais arcos *fill-ins* são criados apenas entre os nós adjacentes ao nó eliminado (pivoteado). Na verdade forma-se um sub-grafo completo entre estes nós (clique).

Cabe observar que o método de *Tinney 1* não leva em consideração os grafos residuais. Por esta razão ele é raramente usado preferindo-se a abordagem denominada *minimum degree* (ou Esquema 2), discutido a seguir.

Minimum Degree (Esquema 2)

Na reordenação por *minimum degree*, a cada passo da eliminação, escolhe-se, como próximo nó a ser eliminado, o nó de menor grau do grafo residual. Se houver empate, a escolha entre eles é arbitrária (Tinney e Walker [92]).

O funcionamento deste esquema exige que se conheça, a cada passo, os valores atualizados dos graus dos nós que eventualmente tenham sido atingidos pelas sucessivas eliminações.

Um ponto importante a ser observado na implementação do *minimum degree*, é que a eliminação de qualquer nó só afeta os graus dos nós pertencentes a sua adjacência.

O enunciado deste esquema em termos matriciais seria:

Numere as linhas de forma que, a cada passo do processo de eliminação, a próxima linha a ser processada é aquela que apresente o menor número de elementos não nulos fora da diagonal. Se houver empate, escolhe-se qualquer uma.

Minimum Local Fill-in (Esquema 3)

A cada passo da eliminação, elimina-se o nó de menor valência do grafo residual. Em caso de empate, escolha entre eles arbitrariamente (Tinney e Walker [92]).

O esforço computacional exigido por este método é bastante superior ao exigido pelo método *minimum degree*.

Em termos matriciais, este esquema pode ser enunciado da seguinte forma:

Numere as linhas de maneira que, a cada passo da eliminação, a próxima linha a ser processada seja aquela que introduz o menor número de elementos não nulos (*fill-ins*). Se mais de uma linha satisfizer esta condição, escolha arbitrariamente.

Na prática ambos os métodos (*minimum degree* e *minimum local fill-in*) são implementados fazendo uma simulação da estratégia de ordenamento, a qual consiste em fazer a ordenação antes e não durante o processo de decomposição triangular (eliminação gaussiana). Para que isto seja possível, é necessário implementar algoritmos que simulem as eliminações, sem efetivamente executá-las.

Nested Dissection (Examinar parte por parte)

Existem outros métodos de ordenação das linhas e colunas da matriz Q, como, por exemplo, o *nested dissection*, usado em implementações paralelas (George, Heath, Liu e NG [40]).

3.4.5 Estrutura de Dados

Para uma implementação eficiente dos métodos de pontos interiores, devemos primeiro escolher uma estrutura de dados conveniente para armazenar as informações do problema de fluxo a custo mínimo. Tal estrutura de dados deve ser flexível e tão simples quanto possível; não deve usar muito espaço em memória e deve armazenar somente as entradas não nulas da matriz A. No presente trabalho nós usamos uma estrutura de dados apropriada para redes (Kennington e Helgason [58]), a qual consiste em armazenar a matriz de incidência A em dois vetores de tamanho n, denominados cauda $t = (t_j)$ e cabeça $h = (h_j)$

dos arcos. Além disto, armazena-se o vetor de demandas $b = (b_i)$, de m componentes, e os vetores de custo $c = (c_j)$ e capacidade $d = (d_j)$, ambos com n componentes.

Para ilustrar a apresentação da implementação computacional dos métodos de pontos interiores usando a estrutura de dados definida acima, consideremos o seguinte exemplo de problema de fluxo de custo mínimo:

$$
\begin{array}{ll}
\textit{Minimizar} & \begin{pmatrix} 2 & 1 & 3 & 1 & 1 & 5 & 6 \end{pmatrix} \; x \\
\textit{Sujeito a} & \begin{pmatrix}
1 & 1 & & & & & \\
-1 & & 1 & 1 & & & \\
 & -1 & & & 1 & & \\
 & & -1 & & & 1 & \\
 & & & -1 & -1 & & 1 \\
 & & & & & -1 & -1
\end{pmatrix} x = \begin{pmatrix} 1 \\ -2 \\ 1 \\ -4 \\ 1 \\ 3 \end{pmatrix} \\
 & 0 \leq (x_1, x_2, x_3, x_4, x_5, x_6, x_7) \leq 5
\end{array}
$$

Neste caso, os vetores (t, h, c, d) = (cauda, cabeça, custo, capacidade), e o vetor b de demandas, são dados a seguir:

t	h	c	d
1	2	2	5
1	3	1	5
2	4	3	5
2	5	1	5
3	5	1	5
4	6	5	5
5	6	6	5

b
1
−2
1
−4
1
3

Observe que ao invés de armazenar as 42 entradas da matriz A (ela tem 6 linhas e 7 colunas) é suficiente armazenar 14 entradas (elementos não nulos da matriz A), os quais estão armazenados nos vetores t e h. Em seguida, deve-se escolher uma estrutura de dados conveniente para armazenar a matriz $Q = AD^*A^T$.

A matriz Q tem a seguinte estrutura,

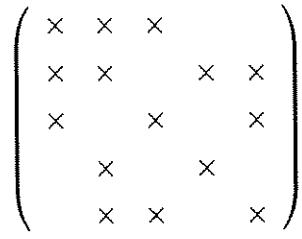


Figura 3.4: Estrutura da matriz Q

cujo grafo associado é ilustrado na Figura 3.5.

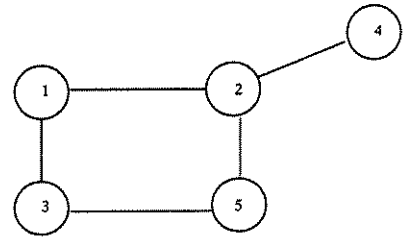


Figura 3.5: Grafo associado à estrutura da matriz Q

Tendo em vista que os métodos de pontos interiores devem resolver a cada iteração um sistema simétrico e positivo definido do tipo $Qy = \hat{b}$, onde $Q = AD^*A^T$, é conveniente armazenar a matriz Q de maneira que sua esparsidade seja bem explorada.

Para armazenar a estrutura da matriz Q foi usada a seguinte estrutura de dados (Knuth [60]):

`ipri[i]` – índice no vetor `ilin` do primeiro elemento não nulo, fora da diagonal, na $i-$ésima coluna. `ipri`$[m+1]$ aponta para a primeira posição disponível para preenchimento.

`ilin[k]` – índice de linha do $k-$ésimo elemento não nulo de Q.

`prox[k]` – índice de `ilin` do próximo elemento não nulo na mesma coluna de Q (ou da lista de posições disponíveis).

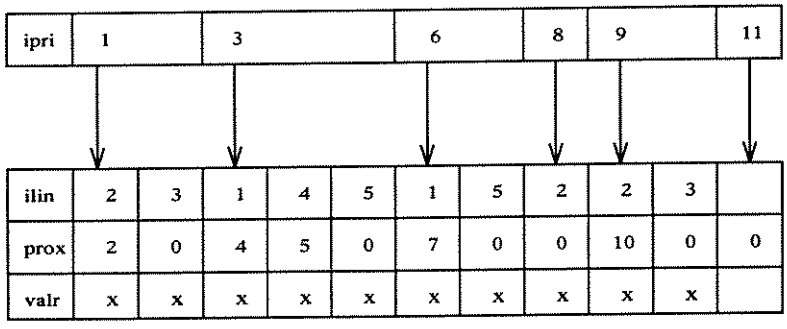
`valr[k]` – valor do elemento não nulo da matriz Q.

`noze[i]` – número de elementos não nulos fora da diagonal na i–ésima coluna.

`nord[i]` – ordem de pivoteamento da i–ésima coluna.

É importante salientar que os vetores acima servem para armazenar a estrutura da matriz Q, como uma estrutura de lista ligada múltipla (Knuth [60]), onde cada coluna é armazenada em uma estrutura de lista ligada simples, contendo apenas os elementos não nulos daquela coluna.

Considerando a estrutura da matriz Q ilustrada na Figura 3.4, inicialmente temos os seguintes vetores (`ipri`, `ilin`, `prox` e `valr`):



Assim, os elementos não nulos que estão na terceira coluna da matriz Q, estão armazenados da seguinte forma `ipri`[3] = 6, significando que o primeiro elemento não nulo da coluna 3 está na posição 6 do vetor `ilin`; como `ilin`[6] = 1, trata-se do elemento (1,3); como `prox`[6] = 7, então o próximo elemento não nulo está na posição 7 e, como `ilin`[7] = 5, está na linha 5 (trata-se do elemento (5,3)). Finalmente, como `prox`[7] = 0, então não existem mais elementos não nulos nessa coluna (coluna 3).Vale lembrar que, os vetores `ipri`, `ilin` e `prox` são utilizados para armazenar os elementos não nulos da matriz Q que estão fora da diagonal, coluna por coluna.

Um outro vetor, denominado `noze` armazena informações sobre o número de elementos não nulos em cada coluna da matriz Q (não incluindo os elementos da diagonal). Por exemplo,

noze	2	3	2	1	2

significa que na matriz Q temos dois elementos não nulos fora da diagonal na coluna 1, três elementos não nulos fora da diagonal na coluna 2, dois elementos não nulos fora da diagonal

na coluna 3, um elemento não nulo fora da diagonal na coluna 4 e dois elementos não nulos fora da diagonal na coluna 5 (vide Figura 3.4).

A próxima seção apresenta a implementação computacional da fatoração simbólica.

3.4.6 Implementação da Fatoração Simbólica

Uma vez escolhida uma estrutura de dados para armazenar a estrutura da matriz Q, devemos aplicar uma estratégia de ordenação para fazer os pivoteamentos.

Discutiremos a implementação dos métodos de ordenação, *minimum degree* (MD) e *minimum local fill-in* (MLF). No primeiro método usa-se os vetores `ipri`, `ilin`, `prox` e `noze`, definidos na seção 3.4.5, além do vetor `nord`, o qual deve fornecer a ordem em que tem que ser feito os pivoteamentos.

3.4.6.1 Minimum Degree

Nesta seção é apresentado o algoritmo *minimum degree* para redes. Este método foi inspirado na relação que existe entre a estrutura da matriz Q e o grafo não dirigido H associado a esta estrutura, isto é, a cada elemento não nulo (i,j) e seu correspondente elemento simétrico (j,i), fora da diagonal na matriz Q, está associada uma aresta no grafo H. Como visto no item 3.4.4, fazer um pivoteamento na matriz Q equivale a remover um nó do grafo H e criar novos arcos para formar um clique (sub-grafo completo), entre os nós adjacentes.

Esquematicamente, seja $H = (\mathcal{N}, \mathcal{E})$, onde $\mathcal{N}$ = conjunto de nós (cardinalidade$(\mathcal{N}) = r$) e $\mathcal{E}$ = conjunto de arestas (cardinalidade$(\mathcal{E}) = s$).

Passo 1. Determine o grau de cada nó, no grafo H, associado à estrutura da matriz Q.

Passo 2. Escolha o nó que tenha o menor grau. Se tiver empate, desempate arbitrariamente.

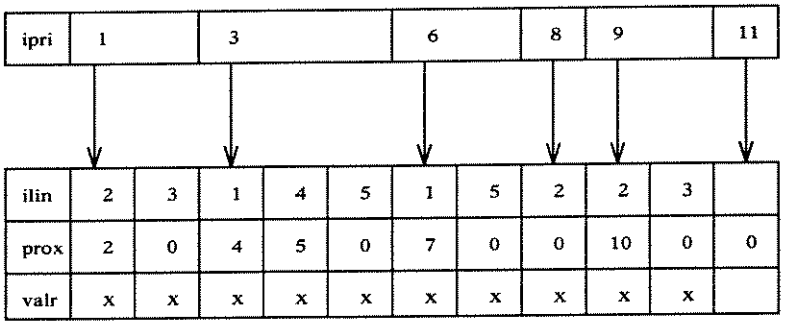
Passo 3. Atualize o grafo H, os conjuntos $\mathcal{N}$ e $\mathcal{E}$, removendo o nó escolhido no passo 2 e todas as arestas que incidem nele; o nó que está sendo removido deve formar um

clique (sub-grafo completo), com seus nós adjacentes. Atualize também os graus dos nós adjacentes ao nó que foi removido.

Passo 4. Se cardinalidade($\mathcal{N}$) = 1, pare. Caso contrário volte ao passo 2.

Usando o exemplo apresentado na seção 3.4.5, pode-se ilustrar o algoritmo *minimum degree*, encontrando uma estratégia de pivoteamento (vetor `nord`), de tal forma que a quantidade de *fill-ins* na matriz PQP^T seja a menor possível, isto é, que preserve a esparsidade da matriz Q.

Neste caso, a estrutura da matriz Q está armazenada na seguinte forma:



o que equivale armazenar as posições dos elementos não nulos da matriz Q (por colunas). Neste exemplo vamos adotar a seguinte representação para os elementos da matriz corrente Q.

$\times$ – *valor original não alterado*
$\otimes$ – *pivô*
$\square$ – *elemento eliminado*
$\bullet$ – *preenchimento*

Inicialmente o vetor `nord` é dado por,

nord	1	2	3	4	5

indicando a seqüência *natural* de pivoteamento.

Aplicando o algoritmo *minimum degree*, temos:

Iteração 1 :

Passo 1. O vetor noze é dado por:

noze	2	3	2	1	2

Ou seja, noze contém o grau de cada nó do grafo H (grafo associado à estrutura da matriz Q), ou seja, indica a quantidade de elementos não nulos fora da diagonal, na $i-$ésima coluna da matriz Q (vide figura 3.4).

Passo 2. Neste caso, o nó que tem o menor grau é o nó 4, pois noze[4] = 1. Logo nord[4] = 1.

Passo 3. O grafo residual é obtido, removendo-se o nó 4 e todas as arestas que incidem nele. Atualizando o vetor noze, temos:

noze	2	2	2		2

Além disto, devemos também atualizar os vetores ipri, ilin e prox, obtendo a nova estrutura de informações ilustradas na Figura 3.6.

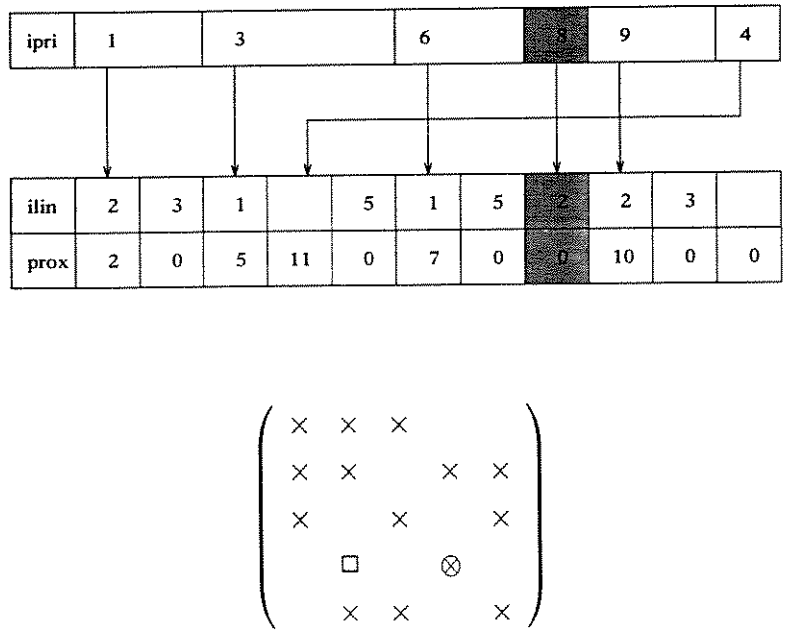


Figura 3.6: Estrutura após o primeiro pivoteamento

Observe que, em relação à estrutura da matriz Q, é como se houvessemos eliminado o elemento que está na posição $(4,2)$, como ilustra a Figura 3.6.

Passo 4. Como cardinalidade($\mathcal{N}$) = 4, voltamos ao passo 2.

Iteração 2 :

Passo 2. Neste caso todos os nós têm o mesmo grau ($grau = 2$), logo pode-se eliminar qualquer um deles. Por exemplo, o nó 1 (menor índice). Logo nord[1] = 2.

Passo 3. O grafo residual é obtido removendo-se o nó 1 e todas as arestas que incidem nele. Atualizando o vetor noze, temos:

noze		2	2		2

Devemos também atualizar os vetores ipri, ilin e prox, obtendo o conjunto de informações representado na Figura 3.7.

ipri	1	6	3	8	9	4

ilin	2	3	2		5	3	5	2	2	3	
prox	2	0	7	11	0	5	0	0	10	0	0

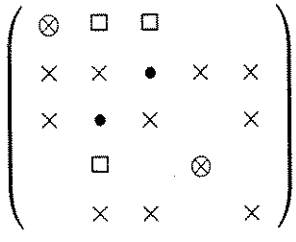


Figura 3.7: Estrutura após o segundo pivoteamento

Observe que, em relação à estrutura da matriz Q, é como se houvessemos eliminado os elementos $(1,2)$ e $(1,3)$. Além disto, foram inseridos o elemento $(3,2)$ na coluna 2 e o elemento $(2,3)$ na coluna 3. Ou seja, existem dois *fill-ins*, como ilustra a Figura 3.7.

Passo 4. Como cardinalidade($\mathcal{N}$) = 3, voltamos ao passo 2.

Iteração 3 :

Passo 2. O grau de cada nó é 2. Assim, pode-se escolher qualquer um deles, por exemplo, o nó 2. Logo `nord`[2] = 3.

Passo 3. O grafo residual é obtido, removendo-se o nó 2 e todas as arestas que incidem nele. Atualizando o vetor `noze`, temos:

noze			1		1

Devemos também atualizar os vetores `ipri`, `ilin` e `prox`, obtendo as informações ilustradas na Figura 3.8.

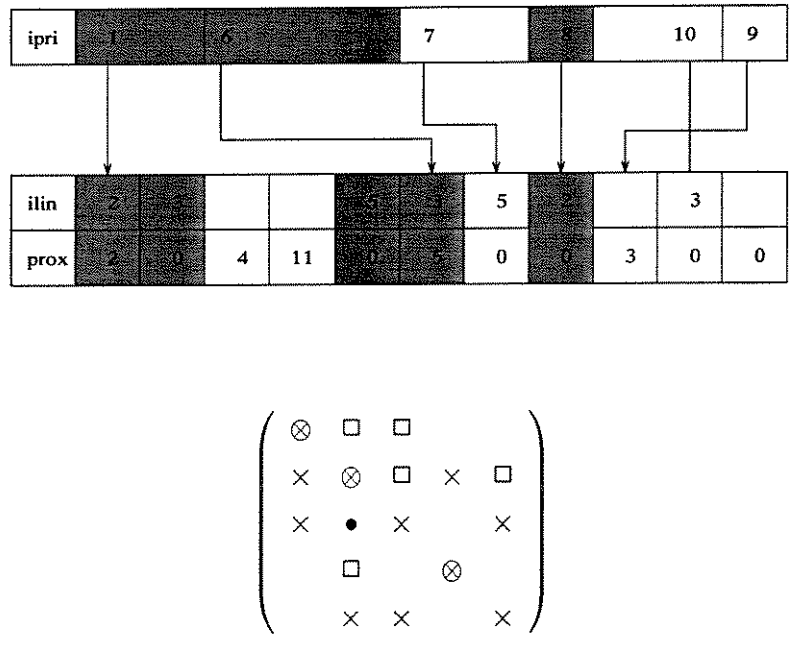


Figura 3.8: Estrutura após o terceiro pivoteamento

Observe que, em ralação à estrutura da matriz Q, é como se houvessemos eliminado os elementos $(2,3)$ e $(2,5)$, como ilustra a Figura 3.8.

Passo 4. Como cardinalidade($\mathcal{N}$) = 2, voltamos ao passo 2.

Iteração 4 :

Passo 2. O grau de todos os nós é 1. Podemos escolher qualquer um deles, por exemplo, o nó 3. Logo `nord`[3] = 4.

Passo 3. O grafo residual é obtido, removendo-se o nó 3 e todas as arestas que incidem nele. Atualizando o vetor `noze`, temos:

noze					0

Atualizando os vetores `ipri`, `ilin` e `prox`, obtemos as informações ilustradas na Figura 3.9.

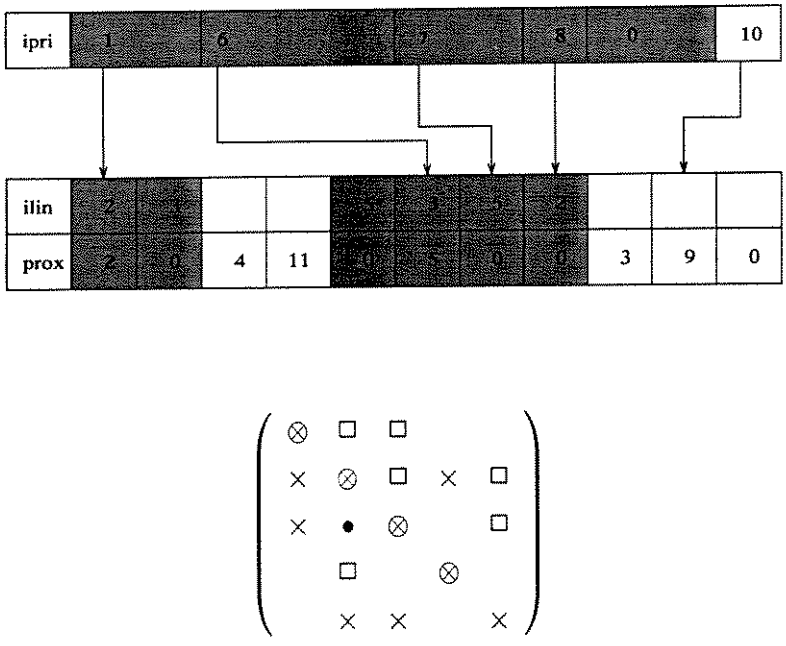


Figura 3.9: Estrutura após o quarto pivoteamento

Com respeito à estrutura da matriz Q, é como se houvessemos eliminado o elemento $(3,5)$, como ilustra a Figura 3.9.

Passo 4. Como cardinalidade($\mathcal{N}$) = 1, pare.

A estrutura acima corresponde, depois de um reordenamento das linhas e colunas de acordo com o vetor `nord`

nord	2	3	4	1	5

à matriz triangular, ilustrada na Figura 3.10, que é o fator de Choleski triangular inferior L. Observe que a primeira linha/coluna desta matriz é a quarta linha/coluna da matriz anterior, pois nord[4] = 1.

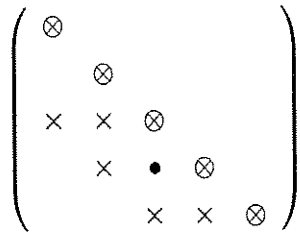


Figura 3.10: Fator de Choleski L

É importante salientar que, ao final da aplicação do algoritmo *minimum degree*, armazena-se apenas a parte triangular inferior da matriz Q (vide Figura 3.10). A simulação dos pivoteamentos nos permite obter como produto da fatoração simbólica a seqüência para execução dos pivoteamentos (dado pelo vetor `nord`) e as informações não nulas que devem ser armazenadas.

3.4.6.2 Minimum Local Fill-in

Neste método, ao invés de usar o vetor `noze`, usa-se o vetor `vale`, o qual armazena a valência de cada nó, sendo a valência de um determinado nó o número de *fill-ins* gerados por esse nó se ele fosse escolhido como pivô. Isto implica que, o esforço computacional para calcular a valência é bem maior que o esforço computacional para calcular o grau de cada nó.

Esquematicamente, seja $H = (\mathcal{N}, \mathcal{E})$, onde $\mathcal{N}$ = conjunto de nós (cardinalidade($\mathcal{N}$) = r) e $\mathcal{E}$ = conjunto de arestas (cardinalidade($\mathcal{E}$) = s).

Passo 1. Determine a valência de cada nó, no grafo H, associado à estrutura da matriz Q.

Passo 2. Escolha o nó que tenha a menor valência. Se tiver empate, desempate arbitrariamente.

Passo 3. Atualize o grafo H, os conjuntos $\mathcal{N}$ e $\mathcal{E}$, removendo o nó escolhido no passo 2 e todas as arestas que incidem nele; o nó que está sendo removido deve formar um clique (sub-grafo completo), com seus nós adjacentes. Atualize também a valência dos outros nós.

Passo 4. Se cardinalidade($\mathcal{N}$) = 1, pare. Caso contrário volte ao passo 2.

Usando o exemplo apresentado na seção 3.4.5, pode-se ilustrar o algoritmo *minimum local fill-in*, encontrando uma estratégia de pivoteamento (vetor `nord`), de tal forma que a quantidade de *fill-ins* na matriz PQP^T seja a menor possível, isto é, que preserve a esparsidade da matriz Q.

Neste particular exemplo, a estrutura da matriz Q e os vetores `ipri`, `ilin`, `prox`, `nord` são idênticos ao do *minimum degree* exceto o vetor `vale` que a cada iteração tem os valores abaixo e determina a mesma ordenação.

inicial:	`vale`	1	3	1	0	1
primeiro pivoteamento:	`vale`	1	1	1		1
segundo pivoteamento:	`vale`		0	0		0
terceiro pivoteamento:	`vale`			0		0
quarto pivoteamento:	`vale`					0

3.5 Métodos Iterativos

Os métodos iterativos, fornecem uma seqüência de pontos y^0, y^1, y^2,... que converge para a solução do sistema $Qy = \hat{b}$, e resolvem este sistema através da solução do seguinte problema equivalente (não linear irrestrito):

$$Minimizar \;\; F(y) = \frac{1}{2} y^T Q y - y^T \hat{b}, \tag{3.10}$$

onde a matriz $Q = AD^*A^T$ é simétrica e positiva definida. Observe que F é uma função quadrática e convexa, com um único ponto de mínimo $y^* = Q^{-1}\hat{b}$.

O problema (3.10) acima é normalmente resolvido usando o método gradiente conjugado (Golub e Van Loan [46]) (o qual consiste em usar direções Q–conjugadas, de forma a minimizar a função F).

A principal vantagem do uso de métodos iterativos é que inexiste o problema de preenchimento (*fill-in*) na matriz AD^*A^T. Uma outra vantagem é que a estrutura de redes do problema pode ser muito bem aproveitada na implementação do algoritmo. No entanto, vale lembrar que os métodos iterativos não permitem a transferência de informações entre iterações do algoritmo de pontos interiores (o que acontece nos métodos diretos onde, por exemplo, a simulação dos pivoteamentos é realizada apenas uma vez no processo de solução).

Seja,

$$Q = AD^*A^T$$

A matriz Q tende a ser mal condicionada ao longo das iterações dos métodos de pontos interiores. Assim, o método gradiente conjugado pré-condicionado é usado para resolver o sistema:

$$(M^{-1}QM^{-1})(My) = M^{-1}\hat{b}$$

onde M é uma matriz positiva definida denominada pré-condicionador.

O objetivo é fazer a matriz $(M^{-1}QM^{-1})$ menos mal condicionada do que a matriz Q e melhorar a convergência do algoritmo gradiente conjugado. Na Figura 3.11 são apresentados os passos do algoritmo gradiente conjugado pré-condicionado (Golub e Van Loan [46]).

Algoritmo Gradiente-Conjugado-Precondicionado(A, D^*, M, $\hat{b}$, y, ϵ)

$$
\begin{array}{lllll}
1 & y_0 & \leftarrow 0 & & \\
2 & r_0 & \leftarrow \hat{b} & & \\
3 & z_0 & \leftarrow M^{-1} r_0 & & \\
4 & p_0 & \leftarrow z_0 & & \\
5 & i & \leftarrow 0 & & \\
6 & \texttt{While} & (\|r_i\|_2^2 & > & \epsilon) \ \texttt{Do} \\
 & 6.1 & q_i & \leftarrow & (AD^*A^T)p_i \\
 & 6.2 & \alpha & \leftarrow & z_i^T r_i / p_i^T q_i \\
 & 6.3 & y_{i+1} & \leftarrow & y_i + \alpha p_i \\
 & 6.4 & r_{i+1} & \leftarrow & r_i - \alpha q_i \\
 & 6.5 & M z_{i+1} & \leftarrow & r_{i+1} \\
 & 6.6 & \beta & \leftarrow & z_{i+1}^T r_{i+1} / z_i^T r_i \\
 & 6.7 & p_{i+1} & \leftarrow & z_{i+1} + \beta p_i \\
 & 6.8 & i & \leftarrow & i + 1; \\
7 & y \leftarrow y_i & & &
\end{array}
$$

Figura 3.11: Algoritmo Gradiente Conjugado Pré-condicionado

O maior esforço computacional deste algoritmo aparece nas linhas (3, 6.1 e 6.5). Essas linhas correspondem à multiplicação matriz-vetor (6.1) e à resolução dos sistemas de equações lineares (3 e 6.5). A linha 3 é calculada uma vez, enquanto as linhas (6.1 e 6.5) são calculadas a cada iteração do algoritmo gradiente conjugado. A multiplicação matriz-vetor é da forma AD^*A^Tp e pode ser calculada sem determinar AD^*A^T, explicitamente. Uma maneira muito simples de calcular este produto matriz-vetor é decompor em três multiplicações matriz-vetor.

Fazendo $u = A^Tp$, $v = D^*u$ e $w = Av$.

Primeiro, calcula-se: $u = A^Tp$

A seguir, calcula-se: $v = D^*u$

Finalmente, calcula-se: $w = Av$

Considerando que, a matriz A, é a matriz de incidência associada ao problema de fluxo de custo mínimo, as entradas não nulas desta matriz podem ser armazenadas em dois vetores de tamanho n, $t = (t_j)$ e $h = (h_j)$.

O cálculo dos vetores u, v e w, é efetuado da seguinte forma:

$$
\begin{array}{llll}
\texttt{for} & i = 1 \ \texttt{to} \ m \ \texttt{do} & w_i \leftarrow 0 & \\
\texttt{for} & j = 1 \ \texttt{to} \ n \ \texttt{do} & & \\
 & aux \leftarrow p[t_j] - p[h_j] & [C\acute{a}lculo\ de\ u] & \\
 & aux \leftarrow aux * d_j & [C\acute{a}lculo\ de\ v] & \\
 & w[t_j] \leftarrow w[t_j] + aux & [C\acute{a}lculo\ de\ w] & \\
 & w[h_j] \leftarrow w[h_j] - aux & &
\end{array}
\tag{3.11}
$$

O esforço computacional para calcular u, v e w é $O(n)$, pois:

No cálculo de u são efetuadas n subtrações

No cálculo de v são efetuadas n multiplicações

No cálculo de w são efetuadas n somas e n subtrações

Portanto, em total são efetuadas n somas, $2n$ subtrações e n multiplicações.

A seguir é apresentado como é realizado o cálculo destes três vetores (u, v e w). Suponha os vetores t e h, dados por:

t	h
1	2
1	3
1	4
2	3
4	3

Além disto,

$$
\begin{array}{lll}
D^* \leftarrow I & & (matriz\ identidade) \\
p \leftarrow (1 & 2 \ 3 \ 4)^T & \\
w \leftarrow (0 & 0 \ 0 \ 0)^T &
\end{array}
$$

Usando as fórmulas apresentadas em (3.11), obtemos os vetores u, v e w:

$$\begin{aligned} u &\leftarrow (-1 \quad -2 \quad -3 \quad -1 \quad 1)^T \\ v &\leftarrow (-1 \quad -2 \quad -3 \quad -1 \quad 1)^T \\ w &\leftarrow (-6 \quad 0 \quad 2 \quad 4)^T \end{aligned}$$

Voltando à resolução dos sistemas (3 e 6.5), no algoritmo gradiente conjugado pré-condicionado (vide figura 3.11), deve-se resolver um sistema linear do tipo,

$$Mz = r \tag{3.12}$$

Uma maneira eficiente de resolver este sistema, é usar um pré-condicionador (matriz M), que permita encontrar z com pouco esforço computacional.

Dois pré-condicionadores têm-se mostrado eficientes em implementações computacionais para resolver o sistema $(AD^*A^T)y = \hat{b}$.

3.5.1 Pré-condicionador Diagonal

O pré-condicionador diagonal,

$$M = diag(AD^*A^T)$$

Este pré-condicionador pode ser calculado em $O(n)$ operações e o sistema (3.12) pode ser resolvido em $O(m)$ divisões.

3.5.2 Pré-condicionador Árvore Geradora Máxima

Existe um outro pré-condicionador, denominado pré-condicionador de árvore geradora máxima, que melhora o número de iterações do gradiente conjugado nas iterações finais do método de pontos interiores (afins, primais-duais). Este pré-condicionador foi usado por Resende e Veiga [88], na implementação do método afim-dual para resolver problemas de fluxo de custo mínimo.

Seja M,

$$M = S_k \mathcal{D}_k S_k^T$$

onde S_k é uma submatriz de A formada pelas colunas correspondentes aos arcos que aparecem na solução do problema da árvore geradora máxima, e $\mathcal{D}_k$ é uma matriz diagonal associada a S_k. A árvore geradora máxima é obtida a partir do grafo associado ao problema de fluxo de custo mínimo, cujos pesos nos arcos são as componentes (que estão na diagonal) da matriz de escalamento D^*. Supondo que os arcos t_1, t_2, ..., t_q são os arcos que aparecem na solução da árvore geradora máxima, então:

$$\mathcal{D}_k \quad = \quad diag(1/d_{t_1}, 1/d_{t_2}, ..., 1/d_{t_q})$$

Logo o sistema $Mz = r$, torna-se

$$(S_k \mathcal{D}_k S_k^T) z = r \tag{3.13}$$

Este sistema é resolvido em $O(m)$ operações, tirando proveito da estrutura de dados da árvore geradora máxima.

O sistema (3.13) pode ser resolvido, eficientemente, da seguinte forma.

$$\text{Fazendo} \quad u = \mathcal{D}_k S_k^T z \quad e \quad v = S_k^T z$$

$$\begin{array}{lrcl} \text{Inicialmente, resolve} - \text{se} & S_k u & = & r \\ \text{A seguir calcula} - \text{se} & v & = & \mathcal{D}_k^{-1} u \\ \text{Finalmente, resolve} - \text{se} & S_k^T z & = & v \end{array}$$

O primeiro sistema, $S_k u = r$, é resolvido percorrendo a árvore geradora máxima das folhas à raiz. O vetor v é calculado efetuando-se m divisões. Finalmente, o sistema $S_k^T z = v$ é resolvido percorrendo a árvore geradora máxima da raiz às folhas.

Para resolver o sistema $S_k u = r$, usando os vetores $t = (t_j)$, $h = (h_j)$ e $r = (r_i)$, devemos inicialmente percorrer a árvore geradora máxima a partir de um nó folha.

Para percorrer a árvore a partir de um nó folha, usa-se dois vetores de m componentes:

fior[i] – percorre todos os nós folha e um caminho ascendente até a raiz (a partir do último nó folha encontrado); a primeira componente aponta da raiz para um nó folha.

arci[i] – aponta o arco que incide no nó i ($i \neq m$).

A seguinte figura (Figura 3.12), ilustra os vetores $arci$ e $fior$ na árvore geradora máxima (nó raiz = 4).

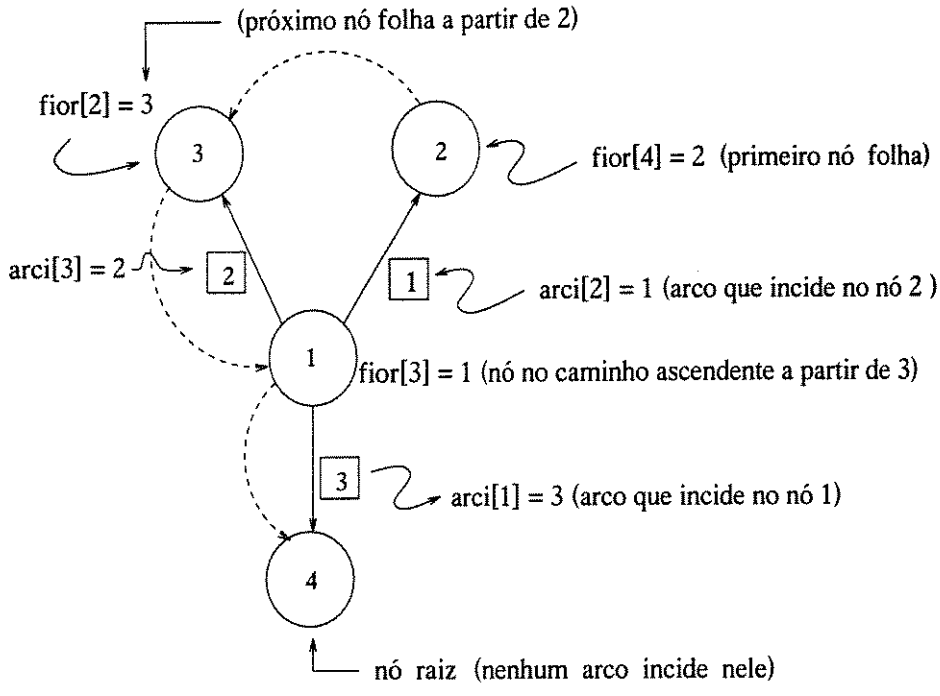


Figura 3.12: Estrutura de dados para percorrer a árvore das folhas à raiz

A seguir é apresentado o algoritmo para resolver o sistema $S_k u = r$ tirando proveito da estrutura de rede.

Algoritmo 1

Seja m o nó raiz

Passo 1 Para $i = 1$ até $m-1$ faça $u[i] \leftarrow r[i]$ (o Passo 1 não inclui o nó raiz m porque não existe arco a partir deste nó).

Passo 2 Percorra a árvore geradora máxima das folhas à raiz.

$f \leftarrow fior[m]$
`While` $(f \neq m)$ `Do`
$j \leftarrow arci[f]$
`if` $(f = t_j)$ `then` $u[h_j] \leftarrow u[h_j] + u_f$
`else` $\{\ u[t_j] \leftarrow u[t_j] + u_f;\ u_f \leftarrow -u_f\ \}$
$f \leftarrow fior[f]$

Para resolver o sistema $S_k^T z = v$, usando os vetores $t = (t_j)$, $h = (h_j)$ e $v = (v_i)$, devemos inicialmente percorrer a árvore geradora máxima a partir do nó raiz.

Para percorrer a árvore a partir do nó raiz, usa-se dois vetores de m componentes:

$fio[i]$ – aponta para o descendente mais próximo de um certo nó, ou para o primeiro descendente do ascendente mais próximo (de um nó folha).

$arci[i]$ – aponta o arco que incide no nó i $(i \neq m)$.

A seguinte figura (Figura 3.13), ilustra os vetores $arci$ e fio na árvore geradora máxima (nó raiz = 4)

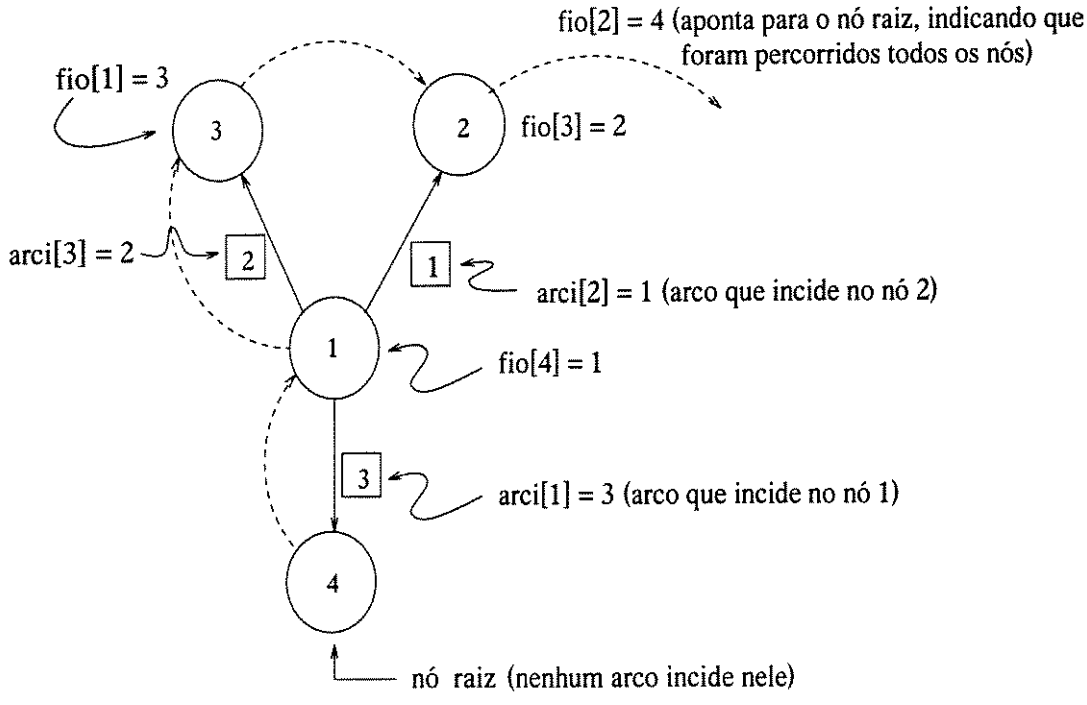


Figura 3.13: Estrutura de dados para percorrer a árvore da raiz às folhas

A seguir é apresentado o algoritmo para resolver o sistema $S_k^T z = v$ tirando proveito da estrutura de rede.

Algoritmo 2

Seja m o nó raiz

Passo 1 Para $i = 1$ até $m - 1$ faça $z[i] \leftarrow v[i]$

Passo 2 Percorra a árvore geradora máxima da raiz às folhas.

$k \leftarrow fio[m]$
`While` $(k \neq m)$ `Do`
$j \leftarrow arci[k]$
`if` $(k = t_j)$ `then` $z_k \leftarrow z_k + z[h_j]$
`else` $\{ z_k \leftarrow -z_k; \quad z_k \leftarrow z_k + z[t_j] \}$
$k \leftarrow fio[k]$

A seguir é apresentado um exemplo ilustrativo para mostrar como funciona o pré-condicionador árvore geradora máxima.

Dados os vetores t e h,

t	h
1	2
1	3
2	3
3	4
2	4

Suponha que na iteração k:

$$D^* \leftarrow diag(8, 6, 5, 4, 9)$$

Inicialmente, devemos encontrar a árvore geradora máxima do grafo associado ao problema de fluxo de custo mínimo, cujos pesos nos arcos são as componentes da matriz de escalamento D^*. Considere, para ilustração o grafo representado na Figura 3.14. A árvore geradora máxima é mostrada na Figura 3.15.

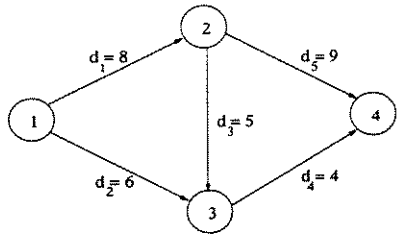


Figura 3.14: Grafo associado ao problema da árvore geradora máxima

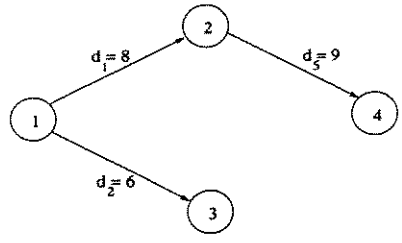


Figura 3.15: Árvore geradora máxima

Portanto, a solução é dada por,

$$d_2 = 6, \quad d_1 = 8 \quad e \quad d_5 = 9$$

Assim, S_k é uma submatriz de A formada pelas colunas 2, 1 e 5, $\mathcal{D}_k \leftarrow diag(\frac{1}{6}, \frac{1}{8}, \frac{1}{9})$, e a solução do sistema $Mz = r$ é obtida na seguinte forma.

Cálculo de u:

Seja $r = (1, 2, 3)^T$ e $r_4 = 0$. Aplicando os passos do Algoritmo 1, temos:

Passo 1 $u \leftarrow (1, 2, 3)^T$

Passo 2 Para todos os arcos ($j = 2$, 1 e 5) que estão na árvore geradora máxima. Faça

$$\begin{aligned} Para \quad j = 2: \quad & u \leftarrow (-3, 4, 2)^T \\ Para \quad j = 1: \quad & u \leftarrow (-3, 4, 6)^T \\ Para \quad j = 5: \quad & u \leftarrow (-3, 4, 6)^T \end{aligned}$$

Observa-se que, neste caso, estamos percorrendo a árvore das folhas (nó 3) à raiz (nó 4), como ilustra a Figura 3.16. Usando-se uma estrutura de dados conveniente é possível identificar o nó folha, como já ilustrado neste item.

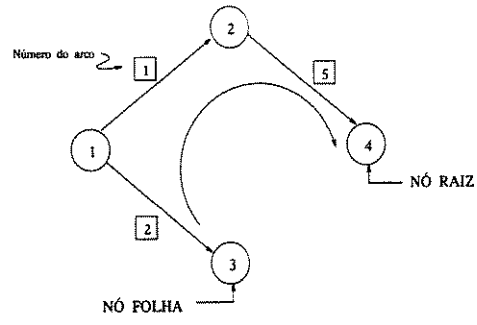


Figura 3.16: Solução do sistema $S_k u = r$

Cálculo de v:

$v \leftarrow \mathcal{D}_k^{-1} u$, então

$$v \leftarrow \begin{pmatrix} 6 & & \\ & 8 & \\ & & 9 \end{pmatrix} \begin{pmatrix} -3 \\ 4 \\ 6 \end{pmatrix} \leftarrow \begin{pmatrix} -18 \\ 32 \\ 54 \end{pmatrix}$$

Cálculo de z:

Aplicando os passos do Algoritmo 2, temos:

Passo 1 $z \leftarrow (-18, 32, 54)^T$

Passo 2 Para todos os arcos ($j = 5$, 1 e 2) que estão na árvore geradora máxima. Faça

$$\begin{aligned} Para \quad j = 5: \quad & z \leftarrow (32, 54, -18)^T \\ Para \quad j = 1: \quad & z \leftarrow (86, 54, -18)^T \\ Para \quad j = 2: \quad & z \leftarrow (86, 54, 104)^T \end{aligned}$$

Observa-se que, neste caso, estamos percorrendo a árvore da raiz (nó 4) às folhas (nó 3), como ilustra a Figura 3.17.

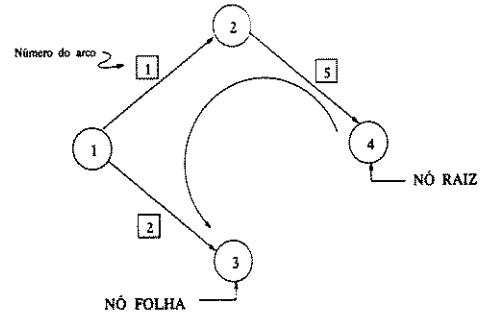


Figura 3.17: Solução do sistema $S_k^T z = v$

O problema da árvore geradora máxima pode fornecer soluções com ciclos, quando existe degenerescência no dual. Neste caso, para se obter o pré-condicionador retira-se do ciclo o arco com menor peso.

Na prática, o pré-condicionador diagonal é efetivo durante as primeiras iterações dos métodos de pontos interiores (afins, primais-duais e preditor-corretor). Ao longo das iterações (após de um certo número delas), o pré-condicionador árvore geradora máxima é mais efetivo, pois ele se torna uma boa aproximação da matriz AD^*A^T. A questão fundamental é em que momento deve-se mudar de pré-condicionador. Adotamos a regra proposta por Golub e Van Loan [46]: se na solução do sistema $Qy = \hat{b}$ o método gradiente conjugado faz mais do que $\lambda\sqrt{m}$ iterações ($\lambda > 0$), deve-se substituir o pré-condicionador. Em nosso caso, faz-se a substituição pelo pré-condicionador árvore geradora máxima — esta regra foi usada em implementações semelhantes por Resende e Veiga [89]. Nas implementações computacionais o pré-condicionador de árvore geradora máxima foi atualizado a cada iteração, a partir do momento em que êle passa a ser adotado.

Com relação ao algoritmo Gradiente Conjugado Pré-condicionado (Figura 3.11), para determinar quando a direção aproximada y_i fornecida pelo Gradiente Conjugado (GC) é satisfatória, pode-se calcular o ângulo entre os vetores $(AD^*A^T)y_i$ e $\hat{b}$. Quando $|1-\cos\theta| < \epsilon$ (nas implementações, atribui-se $\epsilon = 10^{-3}$) [89], considera-se que a solução do sistema foi obtida.

O valor exato de $\cos\theta$ pode ser obtido pela seguinte expressão:

$$\cos\theta = \frac{|\hat{b}^T(AD^*A^T)y_i|}{||\hat{b}||\ ||(AD^*A^T)y_i||}$$

A avaliação exata de $\cos\theta$ tem a complexidade de uma iteração do GC — logo, não é recomendável efetuar este cálculo a cada iteração do algoritmo GC.

Um procedimento mais eficiente para avaliação de $\cos\theta$ segue da observação que y_i é aproxidamente igual a $\hat{b} - r_i$, sendo r_i uma estimativa do resíduo na $i-$ésima iteração do GC (Resende e Veiga [89]). Neste caso, $\cos\theta$ é calculado com a seguinte aproximação:

$$\cos\theta \approx \frac{|\hat{b}(\hat{b} - r_i)|}{||\hat{b}||\ ||\hat{b} - r_i||}$$

Esta estimativa poder ser calculada a cada iteração do método GC.

3.6 Ordenação Ótima de $(AD^*A^T)y = \hat{b}$ Associado a Problemas de Transportes

Nesta seção estuda-se a solução do sistema linear $(AD^*A^T)y = \hat{b}$, quando a matriz A é uma matriz de restrições associado a um problema de transportes. Mostra-se que, neste caso é possível determinar uma ordenação ótima para a matriz AD^*A^T. O resultado também é valido para o problema da designação, uma vez que se trata de um caso particular do problema de transportes.

3.6.1 Apresentação do Problema

Considere m_1 nós origem e m_2 nós destino (considera-se $m_1 > 1$ e $m_2 > 1$ para se evitar dificuldades de ordem trivial). O nó origem i fornece f_i unidades de um determinado item; o nó destino j requer d_j unidades. Supõe-se que $f_i > 0$ e $d_j > 0$. Existe um arco orientado (i, j) ligando cada par de nós origem−destino, ao qual está associado um custo unitário c_{ij}. O problema de transportes consiste em determinar um fluxo factível x_{ij}, de tal forma que o custo total de transporte $\sum\sum c_{ij}x_{ij}$ seja mínimo. Ou seja, o problema de transporte pode ser sintetizado na formulação abaixo.

$$\begin{array}{lll} Minimizar & \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} c_{ij}x_{ij} & \\ Sujeito\ a & \sum_{j=1}^{m_2} x_{ij} = f_i & i = 1, ..., m_1 \\ & \sum_{i=1}^{m_1} x_{ij} = d_j & j = 1, ..., m_2 \\ & x_{ij} \geq 0 & \forall\ (i,j) \end{array}$$

Supõe-se o problema balanceado, isto é, $\sum_{i=1}^{m_1} f_i = \sum_{j=1}^{m_2} d_j$.

A Figura 3.18 ilustra graficamente o problema,

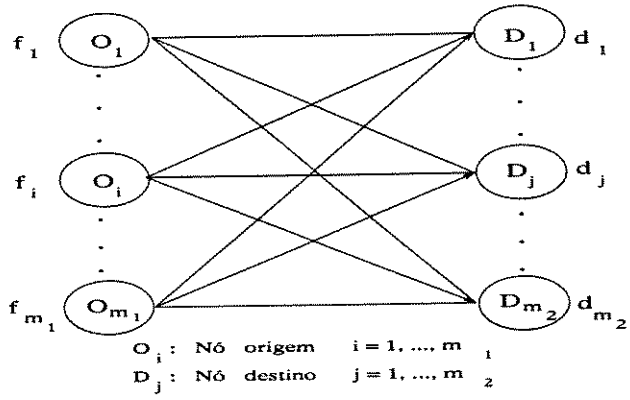


Figura 3.18: O problema de transportes

O grafo associado a um problema de transportes é bipartido, isto é, os nós são particionados em dois conjuntos disjuntos; todos os arcos na rede estão dirigidos de um nó pertencente ao conjunto de nós origem à um nó do conjunto de nós destino. O grafo é também completo, no sentido que existem arcos ligando quaisquer pares de nós origem–destino.

Retirando-se uma equação do conjunto de restrições (escolheu-se a última), para eliminar a redundância, e escrevendo-se o problema de transportes na forma padrão de um problema de programação linear (Bazaraa e outros [11]), tem-se:

$$\begin{aligned} &Minimizar && c^T x \\ &Sujeito\ a && Ax = b \\ & && x \geq 0 \end{aligned}$$

onde a matriz A é a matriz de incidência nó–arco de dimensão $(m_1 + m_2 - 1) \times m_1 m_2$.

3.6.2 Ordenação Ótima

Seja Q,

$$Q = AD^*A^T$$

Pode-se associar à matriz Q um grafo não orientado H com $(m_1 + m_2 - 1)$ nós e $(m_1 m_2 - m_1)$ arestas, representando a localização dos elementos não nulos na matriz (Duff e outros [32]). Os nós do grafo H correspondem às linhas (e colunas) da matriz Q. Existe um arco não orientado j $(j = (i,k) = (k,i), k \neq i)$, quando um elemento fora da diagonal na matriz Q é não nulo $(Q_{ik} = Q_{ki} \neq 0)$.

Estudaremos dois casos: $m_1 > m_2$, e $m_1 \leq m_2$.

CASO 1 $(m_1 > m_2)$

Suponha, por exemplo $m_1 = 5$ e $m_2 = 4$. O grafo H, associado à estrutura da matriz Q, terá 8 nós e 15 arestas. A Figura 3.19 apresenta uma representação esquemática da matriz Q (com um "×" nas posições onde existem elementos não nulos) e do grafo H.

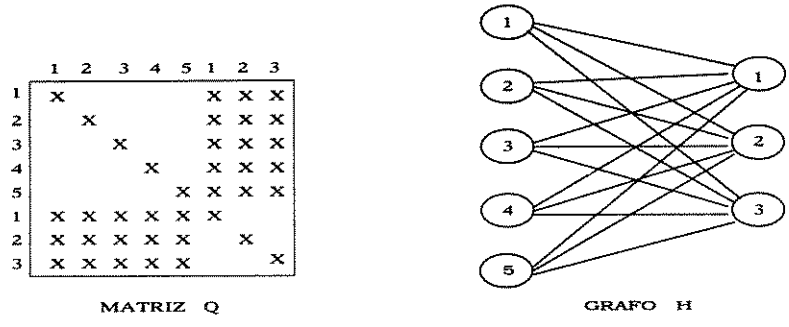


Figura 3.19: Estrutura da matriz Q e seu grafo associado H

Na resolução do sistema, $Qy = \hat{b}$, os pivoteamentos na matriz Q podem ser analisados através do grafo H, considerando as possibilidades a) e b), descritas a seguir.

a) *O pivô é um nó origem.* Suponha que escolhemos como primeiro pivô um elemento correspondente a qualquer nó origem $(1,2,3,4,5)$ — por exemplo pivô $\equiv$ nó 1. Analisando o nó 1 no grafo H, vemos que ele está ligado apenas aos nós destino $(1,2,3)$. Isto significa que na matriz Q teriamos que eliminar os elementos das linhas 6, 7 e 8,

correspondentes aos nós (1,2,3) — os elementos das linhas 2, 3, 4 e 5, abaixo da diagonal, são nulos.

Fazendo o pivoteamento na matriz Q (posição (1,1)), vemos que nas posições (6,7), (6,8), (7,6), (7,8), (8,6) e (8,7) aparecem seis *fill-ins* (posições onde haviam elementos nulos passaram a ser ocupadas por elementos não nulos). Isto pode ser analisado no subgrafo H' de H, formado pelo nó origem 1 (nó pivô) e todos os nós adjacentes a ele (nós destino 1, 2 e 3). Fazer o pivoteamento no nó 1 na matriz Q é equivalente a adicionar novas arestas ao subgrafo H', de tal forma que ele se torne um clique (Duff e outros [32], Tinney e Walker [92]), como ilustrado na Figura 3.20.

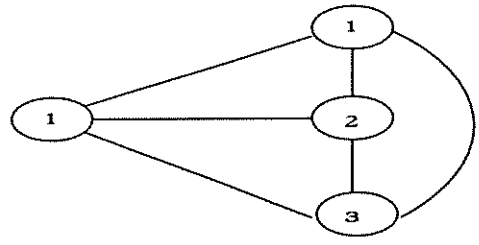


Figura 3.20: Clique formado no subgrafo H'

Em seguida, podemos fazer os pivoteamentos escolhendo uma seqüência qualquer nos nós origem $(2,3,4,5)$ e a seguir nos nós destino $(1,2,3)$, nesta ordem. Nesta seqüência, é facil ver que não serão gerados mais *fill-ins* na matriz Q. Assim, neste caso, o número total de *fill-ins* na matriz L (fator de Choleski da matriz Q) é o número de arestas novas entre os nós destino.

A matriz Q é simétrica e a solução do sistema, $Qy = \hat{b}$, será encontrada através da fatoração de Choleski, $Q = LL^T$. Assim, precisamos considerar apenas os *fill-ins* em uma das matrizes triangulares, L ou sua transposta L^T.

O número de *fill-ins*, dado pelo número de arestas novas em L, com os pivoteamentos na ordem estabelecida acima, pode ser generalizado pela equação (3.14).

$$\begin{pmatrix} m_2 - 1 \\ 2 \end{pmatrix} = \frac{(m_2-1)(m_2-2)}{2} \tag{3.14}$$

No exemplo, como $m_1 = 5$ e $m_2 = 4$, o número total de *fill-ins* gerados é:

$$\begin{pmatrix} 4-1 \\ 2 \end{pmatrix} = \frac{(4-1)(4-2)}{2} = 3$$

A Figura 3.21 representa o grafo G_L, associado à matriz triangular inferior L.

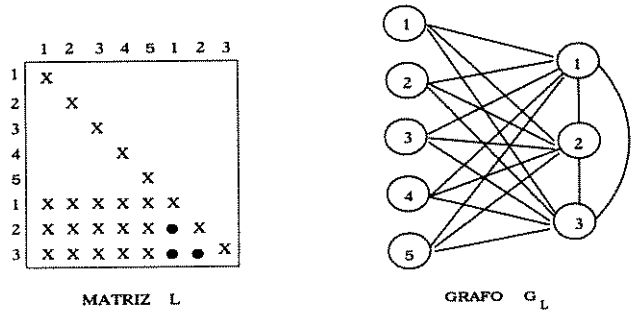


Figura 3.21: Estrutura da matriz L e seu grafo associado G_L

b) O pivô é um nó destino. Suponha agora que escolhemos como primeiro pivô um elemento correspondente a qualquer nó destino (1,2,3) — por exemplo pivô ≡ nó destino 2. Analisando o nó destino 2 no grafo H, vemos que ele está ligado aos nós origem $(1,2,3,4,5)$. Isto pode ser analisado no subgrafo H'' de H, formado pelo nó destino 2 (nó pivô) e todos os nós adjacentes a ele (nós origem 1, 2, 3, 4 e 5). Fazer o pivoteamento no nó destino 2 (nó pivô) na matriz Q é equivalente a adicionar novas arestas ao subgrafo H'', de tal forma que ele se torne um clique (Duff e outros [32], Tinney e Walker [92]), como ilustra a Figura 3.22.

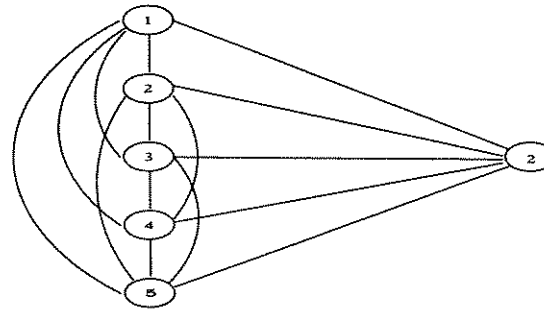


Figura 3.22: Clique formado no subgrafo H''

Em seguida, podemos fazer os pivoteamentos escolhendo uma seqüência qualquer nos nós destino (1,3) e a seguir nos nós origem $(1,2,3,4,5)$, nesta ordem. Neste caso,

é fácil ver que não serão gerados mais *fill-ins* na matriz Q. Assim, o número total de *fill-ins* em L é dado pela equação (3.15).

$$\begin{pmatrix} m_1 \\ 2 \end{pmatrix} = \frac{m_1(m_1-1)}{2} \tag{3.15}$$

No exemplo, como $m_1 = 5$ e $m_2 = 4$, então o número total de *fill-ins* gerados é:

$$\begin{pmatrix} 5 \\ 2 \end{pmatrix} = \frac{5(5-1)}{2} = 10$$

A Figura 3.23 ilustra o grafo G_L, associado à matriz triangular inferior L, após os pivoteamentos na ordem estabelecida acima, e tendo em vista que $PQP^T = LL^T$ (fatoração de Choleski) — a matriz P é uma matriz de permutação que coloca o nó destino 2 na posição adequada (faz com que a linha/coluna 1, correspondente ao nó origem 1, e 7 correspondente ao nó destino 2, sejam permutadas).

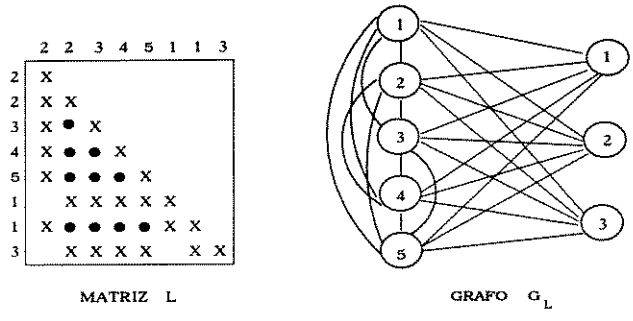


Figura 3.23: Estrutura da matriz L e seu grafo associado G_L

Caso 2 ($m_1 \leq m_2$)

Suponha, por exemplo, $m_1 = 3$ e $m_2 = 5$. O grafo H, associado à estrutura da matriz Q, terá 7 nós e 12 arestas, como ilustra a Figura 3.24.

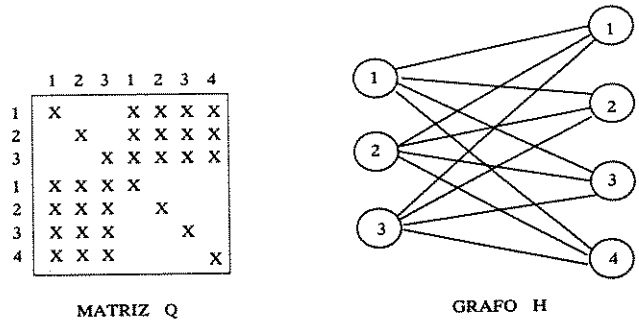


Figura 3.24: Estrutura da matriz Q e seu grafo associado H

Novamente, podemos analisar os pivoteamentos na matriz Q através do Grafo H, considerando as possibilidades a) e b) a seguir.

a) *Pivoteamento em um nó origem.* Suponha que escolhemos como primeiro pivô qualquer nó origem (1,2,3) — por exemplo pivô ≡ nó origem 1. Analisando o nó 1 no grafo H, vemos que ele está ligado aos nós destino (1,2,3,4). Isto significa que na matriz Q teriamos que eliminar os elementos das linhas 4, 5, 6 e 7 (os elementos das linhas 2 e 3, abaixo da diagonal, são nulos).

Fazendo o pivoteamento na matriz Q (posição (1,1)), vemos que nas posições (4,5), (4,6), (4,7), (5,4), (5,6), (5,7), (6,4), (6,5), (6,7), (7,4), (7,5) e (7,6) aparecem doze *fill-ins* (posições onde haviam elementos nulos passaram a ser ocupadas por elementos não nulos).

Analisando no subgrafo H' de H, formado pelo nó origem 1 (nó pivô) e todos os nós adjacentes a ele (nós destino 1, 2, 3 e 4), fazer o pivoteamento no nó origem 1 (nó pivô) na matriz Q é equivalente adicionar novas arestas ao subgrafo H', de tal forma que ele se torne um clique (Duff e outros [32], Tinney e Walker [92]), como ilustra a Figura 3.25.

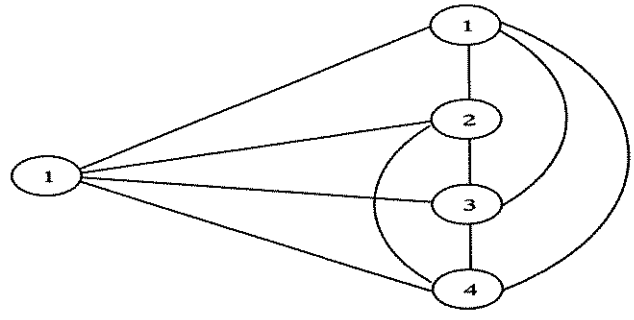


Figura 3.25: Clique formado no subgrafo H'

Em seguida, podemos fazer os pivoteamentos escolhendo uma seqüência qualquer nos nós origem (2,3) e a seguir nos nós destino (1,2,3,4), nesta ordem. Neste caso, vemos que não serão gerados mais *fill-ins* na matriz Q. Assim, o número total de *fill-ins* em L é pela equação (3.16).

$$\begin{pmatrix} m_2 - 1 \\ 2 \end{pmatrix} = \frac{(m_2-1)(m_2-2)}{2} \tag{3.16}$$

No exemplo, como $m_1 = 3$ e $m_2 = 5$, então o número total de *fill-ins* gerados é:

$$\begin{pmatrix} 5 - 1 \\ 2 \end{pmatrix} = \frac{(5-1)(5-2)}{2} = 6$$

A Figura 3.26 ilustra o grafo G_L, associado à matriz triangular inferior L, após os pivoteamentos na ordem estabelecida acima.

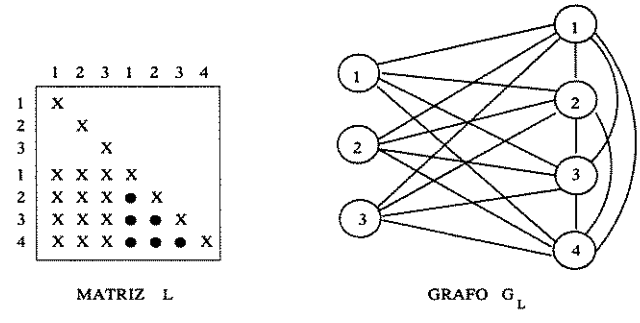


Figura 3.26: Estrutura da matriz L e seu grafo associado G_L

b) Pivoteamento em um nó destino. Suponha agora que escolhemos como primeiro pivô um elemento correspondente a qualquer nó destino (1,2,3,4) — por exemplo pivô $\equiv$ nó destino 4. Analisando o nó destino 4 no grafo H, vemos que ele está ligado aos nós origem (1,2,3). Isto pode ser analisado no subgrafo H'' de H, formado pelo nó destino 4 (nó pivô) e todos os nós adjacentes a ele (nós origem 1, 2 e 3). Fazer o pivoteamento no nó destino 4 (nó pivô) na matriz Q é equivalente a adicionar novas arestas ao subgrafo H'', de tal forma que ele se torne um clique (Duff e outros [32], Tinney e Walker [92]), como mostra a Figura 3.27.

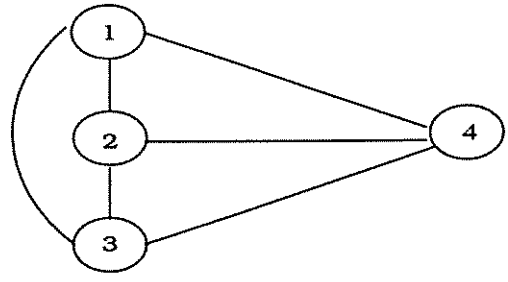


Figura 3.27: Clique formado no subgrafo H''

Em seguida, podemos fazer os pivoteamentos escolhendo uma seqüência qualquer nos nós destino (1,2,3) e a seguir nos nós origem (1,2,3), nesta ordem. Nesta seqüência, é fácil ver que não serão gerados mais *fill-ins* na matriz Q. Assim, o número total de *fill-ins* em L é dado pela equação (3.17).

$$\begin{pmatrix} m_1 \\ 2 \end{pmatrix} = \frac{m_1(m_1-1)}{2} \tag{3.17}$$

No exemplo, como $m_1 = 2$ e $m_2 = 5$, então o número total de *fill-ins* gerados é:

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} = \frac{3(3-1)}{2} = 3$$

A Figura 3.28 ilustra o grafo G_L, associado à matriz triangular inferior L, após os pivoteamentos na ordem estabelecida acima, e tendo em vista que $PQP^T = LL^T$ (Fatoração de Choleski) — a matriz P é uma matriz de permutação que faz com que as linhas/colunas 1 (nó origem 1) e 7 (nó destino 4) sejam permutadas.

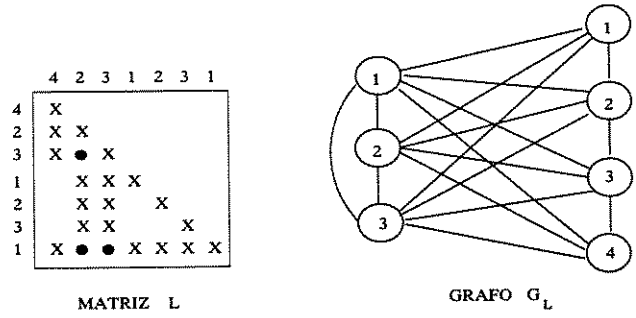


Figura 3.28: Estrutura da matriz L e seu grafo associado G_L

Pelos exemplos discutidos, conclui-se que o número total de *fill-ins* na decomposição é definido pela escolha do primeiro elemento pivô. No **CASO 1** ($m_1 > m_2$) o número de *fill-ins* será dado pela equação (3.14), se o pivô corresponder a um elemento do conjunto de nós origem, e pela equação (3.15), se o pivô corresponder a um elemento do conjunto de nós destino. No **CASO 2** ($m_1 \leq m_2$) o número de *fill-ins* será dado pela equação (3.16), se o pivô corresponder a um elemento do conjunto de nós origem, e pela equação (3.17), se o pivô corresponder a um elemento do conjunto de nós destino. Em ambos os casos, os pivôs subsequentes têm que ser escolhidos de acordo com o primeiro pivô. Ou seja, se o primeiro elemento pivô for um nó origem, então podemos fazer os pivoteamentos escolhendo uma seqüência qualquer nos nós origem e a seguir nos nós destino, nesta ordem. Desta forma obtemos uma seqüência de pivoteamentos que não levará a *fill-ins* adicionais (as seqüências de escolha nos exemplos são apenas indicativas).

Assim, pode-se estabelecer a seguinte proposição.

Proposição. Se A é a matriz de incidência associada a um problema de transportes (ou designação), é possível obter uma ordenação ótima dos nós (no grafo associado à matriz $Q = AD^*A^T$) de tal forma que a quantidade total de *fill-ins* gerados é mínima. Além disto, é possível se determinar a priori a quantidade de *fill-ins* gerados na ordenação ótima — fornecida pela fórmula (3.14), se $m_1 > m_2$, e pela fórmula (3.17), se $m_1 \leq m_2$.

A ordenação ótima é formada pela seqüência de nós $1, 2, ..., m$ ($m = m_1 + m_2$) obtida pela concatenação das seqüências de nós origem e destino da seguinte forma:

- Se $m_1 > m_2$, os nós $1, ..., m_1$ são nós origem e os nós $m_1 + 1, ..., m$ são nós destino.

- Se $m_1 \leq m_2$, os nós $1, ..., m_2$ são nós destino e os nós $m_2 + 1, ..., m$ são nós origem.

É importante salientar que para uma matriz simétrica qualquer a obtenção de uma ordenação ótima é um problema NP−Completo (Yannakakis [98]). Justamente por isso, foram desenvolvidas heurísticas já discutidas neste capítulo, como *minimum degree* e *minimum local fill-in* (Duff e outros [32], Tinney e Walker [92]). No caso do problema de transportes, sendo possível obter ordenações ótimas sem nenhum esforço computacional, estas heurísticas são desnecessárias.

3.7 Estudos de Caso

Esta seção apresenta testes computacionais para solução de sistemas simétricos e positivo definidos do tipo $(AD^*A^T)y = \hat{b}$, para diferentes classes de redes, com diversos graus de esparsidade, usando uma estrutura de dados conveniente à cada método (métodos diretos e iterativos). A matriz D^* adotada nesses estudos foi a matriz de escalamento do método dual-afim.

Os programas foram codificados em linguagem C e executados em estação de trabalho Sun SPARC IPX (Laboratório do DENSIS). Todos os tempos de processamento nas tabelas abaixo estão dados em segundos.

3.7.1 Problemas de Transportes

Neste caso, embora a matriz A seja esparsa (possui somente dois elementos não nulos, $+1$ e -1 em cada coluna), a matriz $Q = AD^*A^T$ é muito densa. No entanto, é possível obter uma ordenação ótima para esta matriz (vide seção 3.6). A tabela 3.1 apresenta um sumário de resultados computacionais para solução por métodos diretos (fatoracão de Choleski com ordenação ótima) e por métodos iterativos (gradiente conjugado, GC, e gradiente conjugado pré-condicionado, PGC, usando o pré-condicionador diagonal).

Nós	Arcos	Choleski	GC	PGC
90	2025	24,90	0,30	0,31
100	2500	33,40	0,40	0,41
200	10000	719,90	3,75	3,81

Tabela 3.1: Tempo de processamento para problemas de transportes

Cabe ressaltar que os resultados da tabela 3.1 mostra os tempos necessários à solução de um único sistema do tipo $(AD^*A^T)y = \hat{b}$. O tempo requerido pelo método Choleski envolve os tempos necessários às fatorações simbólicas e os tempos necessários às fatorações numéricas e resoluções dos sistemas triangulares.

3.7.2 Redes Geradas Aleatoriamente

Neste caso, as redes foram geradas aleatoriamente com graus de esparsidade distintos. As tabelas 3.2 e 3.3 apresentam os resultados obtidos para redes onde a matriz A é muito esparsa (a matriz $Q = AD^*A^T$ é também esparsa). **Choldeg** denota o método de Choleski usando a heurística *minimum degree* para o reordenamento das linhas/colunas; **Cholloc** denota o método de Choleski usando a heurística *minimum local fill-in*; as colunas **PRO 1** mostram os tempos necessários às fatorações simbólicas (simulações dos pivoteamentos); as colunas **PRO 2** mostram a soma dos tempos necessários às resoluções propriamente ditas dos dois sistemas triangulares; a coluna **GC** mostra o tempo gasto pelo método gradiente conjugado, e a última coluna (**PGC**) apresenta o tempo requerido pelo método gradiente conjugado pré-condicionado usando o pré-condicionador diagonal.

Nós	Arcos	Choldeg		Cholloc		GC	PGC
		PRO 1	PRO 2	PRO 1	PRO 2		
90	180	0,93	0,05	5,83	0,05	0,11	0,11
100	200	0,98	0,03	7,35	0,05	0,15	0,13
200	400	7,06	0,10	47,26	0,30	0,38	0,43

Tabela 3.2: Redes aleatórias com número de arcos igual ao dobro de número de nós

Nós	Arcos	Choldeg		Cholloc		GC	PGC
		PRO 1	PRO 2	PRO 1	PRO 2		
90	135	0,66	0,02	2,78	0,02	0,11	0,11
100	150	0,96	0,02	4,28	0,02	0,13	0,15
200	300	6,85	0,06	29,33	0,05	0,38	0,41

Tabela 3.3: Redes aleatórias com número de arcos igual a 1,5 de número de nós

3.8 Comentários

Em linhas gerais, esses primeiros resultados computacionais indicam que, num extremo, os métodos iterativos são mais adequados para redes densas (vide Tabela 3.1). Por outro lado, os métodos diretos (fatoração de Choleski) implementados com heurísticas apropriadas são mais adequados para redes esparsas. Vale lembrar que, os resultados são apenas indicativos, pois a fatoração simbólica (PRO 1) é efetuada uma única vez, enquanto que a fatoração numérica e solução triangular (PRO 2) são efetuadas a cada iteração dos métodos de pontos interiores.

Na simulação dos pivoteamentos usando os métodos diretos (fatoração de Choleski), o método grau mínimo mostrou-se mais eficiente do que o método preenchimento local mínimo (PLM), pois o esforço computacional no método PLM é muito maior.

No método gradiente conjugado foi usado o pré-condicionador diagonal ($M = diag(Q)$). O pré-condicionador árvore geradora máxima não foi utilizado porque sua aplicação se torna vantajosa apenas quando a solução corrente (x) está muito próxima de uma solução básica (o que acontece nas iterações próximas da otimalidade nos métodos de pontos interiores) — situação que não ocorreu nos casos estudados.

É importante salientar que o problema de mal condicionamento pode aparecer logo no início da resolução do sistema $Qy = \hat{b}$. No próximo capítulo serão estudados problemas sem pontos interiores, onde o problema de mal condicionamento torna-se mais visível (na resolução do sistema $Qy = \hat{b}$).

Capítulo 4

Resolução de Problemas com Falta de Pontos Interiores

Situações onde inexistem pontos interiores são freqüentes em problemas de fluxo de custo mínimo. Neste capítulo é apresentado um estudo do comportamento dos métodos de pontos interiores para solução de problemas com essas características. Essencialmente, procura-se responder a seguinte questão:

Será que os métodos de pontos interiores funcionam quando não se dispõe de pontos interiores ?

É importante lembrar que as matrizes de escalamento, D^*, que aparecem nos sistemas $(AD^*A^T)y = \hat{b}$, devem ser diagonais e positivas definidas. Essas propriedades podem não ser satisfeitas em alguns problemas mal comportados.

Vale lembrar a característica da matriz D^* em cada um dos métodos de pontos interiores. Sendo x, y e z pontos interiores factíveis, tem-se:

- para o método afim-primal, $D^* = diag(x_j^2)$;
- para o método afim-dual, $D^* = diag(1/z_j^2)$;
- para os métodos primais-duais, $D^* = diag(x_j/z_j)$.

4.1 Considerações Preliminares

A seguir definiremos o que significa ter um problema primal sem ponto interior e seu correspondente dual (também sem ponto interior).

Considere novamente o par de problemas primal (P) e dual (D):

$$(P)\ \ Minimizar\ \{\ c^T x\ :\ Ax = b,\ x \geq 0\ \}$$

$$(D)\ \ Maximizar\ \{\ b^T y\ :\ A^T y + z = c,\ z \geq 0\ \}$$

Diz-se que o problema primal P possui um ponto (uma solução) factível se e somente se existe um x tal que $Ax = b$, $x \geq 0$. Um ponto factível é interior se $x > 0$, isto é, cada componente do vetor x tem que ser estritamente positiva.

Diz-se que o problema dual D possui um ponto (uma solução) factível se e somente se existe um vetor (y, z) com $z \geq 0$, tal que $A^T y + z = c$. Um ponto factível é interior se $z > 0$, isto é, as variáveis de folga z do problema dual têm que ser estritamente positivas.

Analogamente, pode-se definir uma solução interior factível para os problemas primal e dual canalizados, PC e DC, respectivamente,

$$(PC)\ \ Minimizar\ \{\ c^T x\ :\ Ax = b,\ 0 \leq x \leq d\ \}$$

$$(DC)\ \ Maximizar\ \{\ b^T y - d^T w\ :\ A^T y - w + z = c,\ w, z \geq 0\ \}$$

Diz-se que o problema PC possui um ponto (uma solução) factível se e somente se existe um x tal que $Ax = b$, $0 \leq x \leq d$. Um ponto factível é interior se $0 < x < d$.

Diz-se que o problema DC possui um ponto (uma solução) factível se e somente se existe um vetor (y, w, z) tal que $A^T y - w + z = c$, $w \geq 0$ e $z \geq 0$. Um ponto factível é interior se $w > 0$ e $z > 0$.

É importante salientar que o problema dual canalizado DC sempre admite uma solução interior factível; basta definir, por exemplo, $y_i = 0$ ($\forall i$), e escolher $w = (w_j)$ e $z = (z_j)$ da seguinte forma.

$$\begin{array}{llll} \texttt{se} & (c_j > 0) & \texttt{então} & \{w_j = k; \quad z_j = k + c_j\} \\ \texttt{caso contrário} & (c_j \leq 0) & \texttt{então} & \{w_j = k - c_j; \quad z_j = k\} \end{array}$$

para qualquer valor $k > 0$.

4.2 Algumas Situações de Falta de Pontos Interiores

Nesta seção será estudada a falta de pontos interiores nos problemas primal P e dual D. No item 1 é apresentado um problema (exemplo 1) com soluções ótimas degeneradas, e ausência de soluções interiores primais factíveis. Na seção 2 é apresentado um problema (exemplo 2) com solução ótima dual degenerada e ausência de soluções interiores duais factíveis. Apresenta-se também a resolução desses problemas usando os métodos de pontos interiores afim-primal, afim-dual e primal-dual.

4.2.1 Falta de Pontos Interiores no Problema Primal

O primeiro exemplo (exemplo 1) tem três nós e três arcos.

Formalmente,

$$\begin{array}{lll} \textit{Minimizar} & x_1 + 2x_2 + 3x_3 & \\ \textit{Sujeito a} & \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} \\ & (x_1, x_2, x_3) \geq (0,0,0) \end{array} \tag{4.1}$$

Graficamente o problema (4.1) está representado na Fig. 4.1.

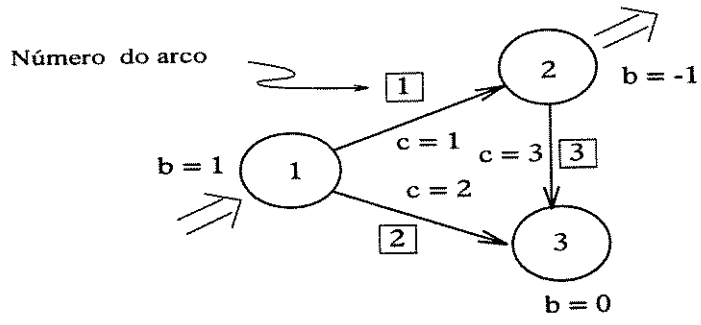


Figura 4.1: Exemplo 1 (3 nós e 3 arcos)

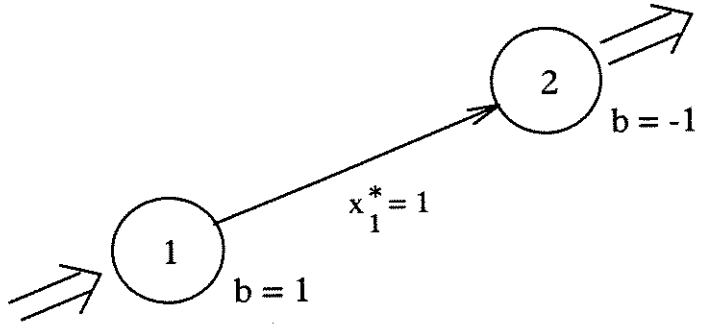


Figura 4.2: Solução ótima do exemplo 1

A solução ótima para o problema (4.1), $x^* = (1,0,0)^T$, está representada na Figura 4.2 (os arcos com fluxos nulos não estão representados na figura). Nota-se que esta solução é primal degenerada, ou seja, existe uma variável básica (x_2 ou x_3) com valor zero. Além disto, o problema possui um único ponto extremo. De fato, fazendo-se operações elementares nas restrições do problema (4.1), obtém-se:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1+x_3 \\ -x_3 \\ x_3 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Logo $x_3 = 0$. Usando-se as restrições originais, conclui-se que $x_1 = 1$ e $x_2 = 0$, confirmando a existência de um único ponto extremo. Consequentemente, não é possível aplicar os métodos de pontos interiores afim-primal, primais-duais e preditor-corretor, pois não existem pontos interiores factíveis.

Para obtermos uma matriz de restrições de posto completo, no exemplo 1, devemos eliminar uma das linhas (por exemplo, a última). O problema (4.1) sem a última

linha torna-se:

$$\begin{array}{lll} \textit{Minimizar} & x_1 + 2x_2 + 3x_3 & \\ \textit{Sujeito a} & \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} & (4.2) \\ & (x_1, x_2, x_3) \geq (0,0,0) & \end{array}$$

Solução pelo Método Afim-Primal

Vamos imaginar a solução deste problema pelo método afim-primal. Para iniciar o processo de solução precisamos de um ponto interior factível. Como vimos, este ponto não existe. Vejamos então as conseqüências para o método de tentarmos iniciar o processo de solução com o único ponto factível, $x = (1,0,0)^T$.

Logo, a matriz de escalamento, D^*, para o método afim-primal, é assim definida:

$$D^* = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

A matriz $Q = AD^*A^T$ é dada por:

$$Q = AD^*A^T = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Para resolver o sistema $Qy = \hat{b}$, devemos encontrar uma matriz triangular inferior L, com elementos estritamente positivos na diagonal, tal que, $LL^T = Q$ (fatoração de Choleski). Neste caso,

$$L = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}$$

A matriz L tem um zero na diagonal ($L_{22} = 0$), ou seja, é singular. Isto significa que as linhas (e colunas) da matriz Q são linearmente dependentes, ou seja, a matriz Q não é de posto completo ($posto(Q) = 1$).

Podemos iniciar o processo de solução com uma aproximação de um ponto inicial interior infactível (um ponto x, tal que $Ax \neq b$, $x > 0$). Isto é, podemos tentar obter uma solução inicial com um procedimento semelhante à fase 1 do simplex, discutido no Capítulo 2. Vejamos o que acontece com este procedimento (fase 1) quando aplicarmos os passos do método afim-primal. Considere o *problema artificial* a ser resolvido na fase 1:

$$
\begin{array}{lcccc}
\textit{Minimizar} & & \lambda & & \\
\textit{Sujeito a} & \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \lambda\,\rho & = & \begin{pmatrix} 1 \\ -1 \end{pmatrix} \\
& (x_1, x_2, x_3) & \geq & (0,0,0) \\
& \lambda & \geq & 0
\end{array}
$$

onde $\rho = b - Ax^0$, $x^0 > 0$ é dado.

Se encontrarmos uma solução (x^k, λ) tal que $\lambda = 0$, então $x = x^k$ é uma solução interior factível de (4.2).

Vejamos os resultados de cada iteração do processo de solução do problema (4.2) com o procedimento afim-primal. Em cada iteração é mostrada a matriz de escalamento, D^*, a matriz $Q = AD^*A^T$, o fator de Choleski (matriz L) associado à matriz Q e o vetor de fluxos x.

$$x_{inicial} = (1.000000 \quad 1.000000 \quad 1.000000)^T$$

ITERAÇÃO 1

$$
\begin{aligned}
D^* &= \begin{pmatrix} 1.000000 & & \\ & 1.000000 & \\ & & 1.000000 \end{pmatrix} \\
Q &= \begin{pmatrix} 2.000000 & -1.000000 \\ -1.000000 & 2.000000 \end{pmatrix} \\
L &= \begin{pmatrix} 1.414214 & 0.000000 \\ -0.707107 & 1.224745 \end{pmatrix} \\
x &= \begin{pmatrix} 1.000000 \\ 0.100000 \\ 0.100000 \end{pmatrix}
\end{aligned}
$$

ITERAÇÃO 2

$$
\begin{aligned}
D^* &= \begin{pmatrix} 1.000000 & & \\ & 0.010000 & \\ & & 0.010000 \end{pmatrix} \\
Q &= \begin{pmatrix} 1.010000 & -1.000000 \\ -1.000000 & 1.010000 \end{pmatrix} \\
L &= \begin{pmatrix} 1.004988 & 0.000000 \\ -0.995037 & 0.141071 \end{pmatrix} \\
x &= \begin{pmatrix} 1.000000 \\ 0.010000 \\ 0.010000 \end{pmatrix}
\end{aligned}
$$

ITERAÇÃO 3

$$
\begin{aligned}
D^* &= \begin{pmatrix} 1.000000 & & \\ & 0.000100 & \\ & & 0.000100 \end{pmatrix} \\
Q &= \begin{pmatrix} 1.000100 & -1.000000 \\ -1.000000 & 1.000100 \end{pmatrix} \\
L &= \begin{pmatrix} 1.000050 & 0.000000 \\ -0.999950 & 0.014142 \end{pmatrix} \\
x &= \begin{pmatrix} 1.000000 \\ 0.001000 \\ 0.001000 \end{pmatrix}
\end{aligned}
$$

ITERAÇÃO 4

$$
\begin{aligned}
D^* &= \begin{pmatrix} 1.000000 & & \\ & 0.000001 & \\ & & 0.000001 \end{pmatrix} \\
Q &= \begin{pmatrix} 1.000001 & -1.000000 \\ -1.000000 & 1.000001 \end{pmatrix} \\
L &= \begin{pmatrix} 1.000000 & 0.000000 \\ -1.000000 & 0.001414 \end{pmatrix} \\
x &= \begin{pmatrix} 1.000000 \\ 0.000100 \\ 0.000100 \end{pmatrix}
\end{aligned}
$$

ITERAÇÃO 5

$$
\begin{aligned}
D^* &= \begin{pmatrix} 1.000000 & & \\ & 0.000000 & \\ & & 0.000000 \end{pmatrix} \\
Q &= \begin{pmatrix} 1.000000 & -1.000000 \\ -1.000000 & 1.000000 \end{pmatrix} \\
L &= \begin{pmatrix} 1.000000 & 0.000000 \\ -1.000000 & 0.000141 \end{pmatrix} \\
x &= \begin{pmatrix} 1.000000 \\ 0.000010 \\ 0.000010 \end{pmatrix}
\end{aligned}
$$

ITERAÇÃO 6

$$D^* = \begin{pmatrix} 1.000000 & & \\ & 0.000000 & \\ & & 0.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1.000000 & -1.000000 \\ -1.000000 & 1.000000 \end{pmatrix}$$

$$L = \begin{pmatrix} 1.000000 & 0.000000 \\ -1.000000 & 0.000014 \end{pmatrix}$$

$$x = \begin{pmatrix} 1.000000 \\ 0.000001 \\ 0.000001 \end{pmatrix}$$

ITERAÇÃO 7

$$D^* = \begin{pmatrix} 1.000000 & & \\ & 0.000000 & \\ & & 0.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1.000000 & -1.000000 \\ -1.000000 & 1.000000 \end{pmatrix}$$

$$L = \begin{pmatrix} 1.000000 & 0.000000 \\ -1.000000 & 0.000001 \end{pmatrix}$$

$$x = \begin{pmatrix} 1.000000 \\ 0.000000 \\ 0.000000 \end{pmatrix}$$

ITERAÇÃO 8

$$
\begin{aligned}
D^* &= \begin{pmatrix} 1.000000 & & \\ & 0.000000 & \\ & & 0.000000 \end{pmatrix} \\
Q &= \begin{pmatrix} 1.000000 & -1.000000 \\ -1.000000 & 1.000000 \end{pmatrix} \\
L &= \begin{pmatrix} 1.000000 & 0.000000 \\ -1.000000 & 0.000000 \end{pmatrix} \\
x &= \begin{pmatrix} 1.000000 \\ 0.000000 \\ 0.000000 \end{pmatrix}
\end{aligned}
$$

Vemos que o elemento L_{22} (da matriz L), vai diminuindo no decorrer das iterações, até se *anular* (para uma certa precisão da máquina). Isto acontece porque a segunda e terceira componentes da diagonal da matriz de escalamento D^* aproximam-se de zero, na busca de atingir a factibilidade. Ou seja, a matriz $Q = AD^*A^T$ *perde posto.* Consequentemente, o algoritmo não pode convergir a um ponto interior factível (lembre-se que o problema não possui pontos interiores).

Veremos no item 4.4 que é possível contornar o problema de *perda de posto* através de um processo de *filtragem* da matriz L.

Solução pelo Método Afim-Dual

Vejamos o que acontece se aplicarmos o método afim-dual para resolver o problema (4.2). O dual de (4.2) é dado por:

$$
\begin{aligned}
&Maximizar && y_1 - y_2 \\
&Sujeito\ a && \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \\
& && (z_1, z_2, z_3) \geq (0,0,0)
\end{aligned}
\qquad (4.3)
$$

Cuja solução ótima é,

$$y^* = (1+\alpha, \alpha)^T \quad e \quad z^* = (0, 1-\alpha, 3-\alpha)^T, \quad \alpha \leq 1$$

ou seja, este problema tem múltiplas soluções ótimas.

Podemos iniciar o processo de solução com uma aproximação de um ponto inicial interior infactível (um ponto (y, z), tal que $A^T y + z \neq c$, $z > 0$). Isto é, podemos tentar obter uma solução inicial com um procedimento semelhante à fase 1 do simplex, discutido no capítulo 2. Vejamos o que acontece com este procedimento (fase 1) quando aplicarmos os passos do método afim-dual. Considere o *problema artificial* a ser resolvido na fase 1:

$$\begin{array}{lcccccccc} \textit{Minimizar} & & & & \lambda & & & & \\ \textit{Sujeito a} & \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} & + & \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} & + & \lambda\rho & = & \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \\ & & (z_1, z_2, z_3) & \geq & (0,0,0) & & & & \\ & & \lambda & \geq & 0 & & & & \end{array}$$

onde $\rho = c - A^T y^0 - z^0$, $z^0 > 0$.

Se encontrarmos uma solução (y^k, z^k, λ) tal que $\lambda = 0$, então (y^k, z^k) é uma solução interior factível de (4.3).

Vejamos os resultados de cada iteração do processo de solução do problema (4.3) com o procedimento afim-dual. Em cada iteração é mostrada a matriz de escalamento, D^*, a matriz $Q = AD^*A^T$, o fator de Choleski (matriz L) associado à matriz Q e os vetores (y, z).

$$\begin{aligned} y_{inicial} &= (0.000000 \quad 0.000000)^T \\ z_{inicial} &= (1.000000 \quad 1.000000 \quad 1.000000)^T \end{aligned}$$

ITERAÇÃO 1

$$D^* = \begin{pmatrix} 1.000000 & & \\ & 1.000000 & \\ & & 1.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 2.000000 & -1.000000 \\ -1.000000 & 2.000000 \end{pmatrix}$$

$$L = \begin{pmatrix} 1.414214 & 0.000000 \\ -0.707107 & 1.224745 \end{pmatrix}$$

$$y = \begin{pmatrix} 1.333333 \\ 0.666667 \end{pmatrix} \quad z = \begin{pmatrix} 1.333333 \\ 0.666667 \\ 1.333333 \end{pmatrix}$$

ITERAÇÃO 2

$$D^* = \begin{pmatrix} 0.562500 & & \\ & 2.250000 & \\ & & 0.562500 \end{pmatrix}$$

$$Q = \begin{pmatrix} 2.812500 & -0.562500 \\ -0.562500 & 1.125000 \end{pmatrix}$$

$$L = \begin{pmatrix} 1.677051 & 0.000000 \\ -0.335410 & 0.006231 \end{pmatrix}$$

$$y = \begin{pmatrix} 1.598667 \\ 0.605333 \end{pmatrix} \quad z = \begin{pmatrix} 0.006667 \\ 0.401333 \\ 2.394667 \end{pmatrix}$$

ITERAÇÃO 3

$$D^* = \begin{pmatrix} 22500.000000 & & \\ & 6.208541 & \\ & & 0.174385 \end{pmatrix}$$

$$Q = \begin{pmatrix} 122506.208541 & -22500.000000 \\ -22500.000000 & 22500.174385 \end{pmatrix}$$

$$L = \begin{pmatrix} 150.020694 & 0.000000 \\ -1490.979309 & 2.526106 \end{pmatrix}$$

$$y = \begin{pmatrix} 1.598848 \\ 0.598881 \end{pmatrix} \quad z = \begin{pmatrix} 0.000033 \\ 0.401152 \\ 2.401119 \end{pmatrix}$$

ITERAÇÃO 4

$$D^* = \begin{pmatrix} 900000000.000198 & & \\ & 6.214152 & \\ & & 0.173449 \end{pmatrix}$$

$$Q = \begin{pmatrix} 900000006.214350 & -900000000.000198 \\ -900000000.000198 & 900000000.173648 \end{pmatrix}$$

$$L = \begin{pmatrix} 30000.000104 & 0.000000 \\ -29999.999896 & 2.527370 \end{pmatrix}$$

$$y = \begin{pmatrix} 1.598849 \\ 0.598849 \end{pmatrix} \quad z = \begin{pmatrix} 0.000000 \\ 0.401151 \\ 2.401151 \end{pmatrix}$$

ITERAÇÃO 5

$$D^* = \begin{pmatrix} 35999999989935.984375 & & \\ & 6.214180 & \\ & & 0.173445 \end{pmatrix}$$

$$Q = \begin{pmatrix} 35999999989942.195312 & -35999999989935.984375 \\ -35999999989935.984375 & 35999999989936.156250 \end{pmatrix}$$

$$L = \begin{pmatrix} 5999999.999162 & 0.000000 \\ -5999999.999161 & 2.527969 \end{pmatrix}$$

$$y = \begin{pmatrix} 1.598849 \\ 0.598849 \end{pmatrix} \quad z = \begin{pmatrix} 0.000000 \\ 0.401151 \\ 2.401151 \end{pmatrix}$$

Vemos que o método afim-dual não tem problemas de convergência. Portanto, se o problema é primal degenerado sem pontos interiores é aconselhável aplicar o método afim-dual. É importante frisar que o método afim-dual trabalha com a formulação em forma de desigualdade, pois as variáveis de folga duais (vetor z) são calculadas implicitamente, usando-se a expressão $z = c - A^T y$; isso torna o método afim-dual mais robusto.

Solução pelo Método Primal-Dual

Vejamos o que contece se aplicarmos os métodos primais-duais para resolver o problema (4.2) e seu dual (problema 4.3).

Podemos iniciar o processo de solução com um ponto (x, y, z) tal que $x > 0$ e $z > 0$. Ou seja, com um ponto interior primal-dual infactível.

Vejamos os resultados de cada iteração do processo de solução do problema (4.3) com o procedimento primal-dual. Em cada iteração é mostrada a matriz de escalamento, D^*, a matriz $Q = AD^*A^T$, o fator de Choleski (matriz L) associado à matriz Q e os vetores (x, y, z).

$$x_{inicial} = (1.000000 \quad 1.000000 \quad 1.000000)^T$$
$$y_{inicial} = (0.000000 \quad 0.000000)^T$$
$$z_{inicial} = (1.000000 \quad 1.000000 \quad 1.000000)^T$$

ITERAÇÃO 1

$$D^* = \begin{pmatrix} 1.000000 & & \\ & 1.000000 & \\ & & 1.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 2.000000 & -1.000000 \\ -1.000000 & 2.000000 \end{pmatrix}$$

$$L = \begin{pmatrix} 1.414214 & 0.000000 \\ -0.707107 & 1.224745 \end{pmatrix}$$

$$x = \begin{pmatrix} 0.692462 \\ 0.615577 \\ 0.000500 \end{pmatrix}$$

$$y = \begin{pmatrix} 0.777778 \\ 0.888889 \end{pmatrix} \quad z = \begin{pmatrix} 1.111111 \\ 1.222222 \\ 2.111111 \end{pmatrix}$$

ITERAÇÃO 2

$$D^* = \begin{pmatrix} 0.623215 & & \\ & 0.503654 & \\ & & 0.000237 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1.126869 & -0.623215 \\ -0.623215 & 0.623452 \end{pmatrix}$$

$$L = \begin{pmatrix} 1.061541 & 0.000000 \\ -0.587086 & 0.527999 \end{pmatrix}$$

$$x = \begin{pmatrix} 1.014685 \\ 0.000308 \\ 0.029677 \end{pmatrix}$$

$$y = \begin{pmatrix} 0.656492 \\ -0.342952 \end{pmatrix} \quad z = \begin{pmatrix} 0.000556 \\ 1.343508 \\ 3.342952 \end{pmatrix}$$

ITERAÇÃO 3

$$D^* = \begin{pmatrix} 1826.432171 & & \\ & 0.000229 & \\ & & 0.008877 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1826.432400 & -1826.432171 \\ -1826.432171 & 1826.441048 \end{pmatrix}$$

$$L = \begin{pmatrix} 42.736780 & 0.000000 \\ -42.736775 & 0.095428 \end{pmatrix}$$

$$x = \begin{pmatrix} 0.997926 \\ 0.004162 \\ 0.000015 \end{pmatrix}$$

$$y = \begin{pmatrix} -0.113270 \\ -1.106689 \end{pmatrix} \quad z = \begin{pmatrix} 0.006581 \\ 2.113270 \\ 4.106689 \end{pmatrix}$$

ITERAÇÃO 4

$$D^* = \begin{pmatrix} 151.641034 & & \\ & 0.001969 & \\ & & 0.000004 \end{pmatrix}$$

$$Q = \begin{pmatrix} 151.643004 & -151.641034 \\ -151.641034 & 151.641038 \end{pmatrix}$$

$$L = \begin{pmatrix} 12.314341 & 0.000000 \\ -12.314181 & 0.044419 \end{pmatrix}$$

$$x = \begin{pmatrix} 1.000077 \\ 0.000002 \\ 0.000156 \end{pmatrix}$$

$$y = \begin{pmatrix} -0.355421 \\ -1.354768 \end{pmatrix} \quad z = \begin{pmatrix} 0.000653 \\ 2.355421 \\ 4.354768 \end{pmatrix}$$

ITERAÇÃO 5

$$D^* = \begin{pmatrix} 1530.777567 & & \\ & 0.000001 & \\ & & 0.000036 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1530.777568 & -1530.777567 \\ -1530.777567 & 1530.777602 \end{pmatrix}$$

$$L = \begin{pmatrix} 39.125153 & 0.000000 \\ -39.125153 & 0.006057 \end{pmatrix}$$

$$x = \begin{pmatrix} 0.999988 \\ 0.000023 \\ 0.000000 \end{pmatrix}$$

$$y = \begin{pmatrix} -1.543770 \\ -2.543703 \end{pmatrix} \quad z = \begin{pmatrix} 0.000067 \\ 3.543770 \\ 5.543703 \end{pmatrix}$$

ITERAÇÃO 6

$$D^* = \begin{pmatrix} 14985.681390 & & \\ & 0.000007 & \\ & & 0.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 14985.681397 & -14985.681390 \\ -14985.681390 & 14985.681390 \end{pmatrix}$$

$$L = \begin{pmatrix} 122.416018 & 0.000000 \\ -122.416018 & 0.002578 \end{pmatrix}$$

$$x = \begin{pmatrix} 1.000001 \\ 0.000000 \\ 0.000001 \end{pmatrix}$$

$$y = \begin{pmatrix} -2.007877 \\ -3.007870 \end{pmatrix} \quad z = \begin{pmatrix} 0.000007 \\ 4.007877 \\ 6.007870 \end{pmatrix}$$

ITERAÇÃO 7

$$D^* = \begin{pmatrix} 150017.707243 & & \\ & 0.000000 & \\ & & 0.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 150017.707243 & -150017.707243 \\ -150017.707243 & 150017.707243 \end{pmatrix}$$

$$L = \begin{pmatrix} 387.321194 & 0.000000 \\ -387.321194 & 0.000439 \end{pmatrix}$$

$$x = \begin{pmatrix} 1.000000 \\ 0.000000 \\ 0.000000 \end{pmatrix}$$

$$y = \begin{pmatrix} -3.445209 \\ -4.445208 \end{pmatrix} \quad z = \begin{pmatrix} 0.000001 \\ 5.445209 \\ 7.445208 \end{pmatrix}$$

ITERAÇÃO 8

$$D^* = \begin{pmatrix} 1499989.837811 & & \\ & 0.000000 & \\ & & 0.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1499989.837811 & -1499989.837811 \\ -1499989.837811 & 1499989.837811 \end{pmatrix}$$

$$L = \begin{pmatrix} 1224.740723 & 0.000000 \\ -1224.740723 & 0.000161 \end{pmatrix}$$

$$x = \begin{pmatrix} 1.000000 \\ 0.000000 \\ 0.000000 \end{pmatrix}$$

$$y = \begin{pmatrix} -4.258089 \\ -5.258089 \end{pmatrix} \quad z = \begin{pmatrix} 0.000000 \\ 6.258089 \\ 8.258089 \end{pmatrix}$$

ITERAÇÃO 9

$$D^* = \begin{pmatrix} 15000011.034830 & & \\ & 0.000000 & \\ & & 0.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 15000011.034830 & -15000011.034830 \\ -15000011.034830 & 15000011.034830 \end{pmatrix}$$

$$L = \begin{pmatrix} 3872.984771 & 0.000000 \\ -3872.984771 & 0.000043 \end{pmatrix}$$

$$x = \begin{pmatrix} 1.000000 \\ 0.000000 \\ 0.000000 \end{pmatrix}$$

$$y = \begin{pmatrix} -5.263421 \\ -6.263421 \end{pmatrix} \quad z = \begin{pmatrix} 0.000000 \\ 7.263421 \\ 9.263421 \end{pmatrix}$$

ITERAÇÃO 10

$$D^* = \begin{pmatrix} 149999992.410071 & & \\ & 0.000000 & \\ & & 0.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 149999992.410071 & -149999992.410071 \\ -149999992.410071 & 149999992.410071 \end{pmatrix}$$

$$L = \begin{pmatrix} 12247.448404 & 0.000000 \\ -12247.448404 & 0.000000 \end{pmatrix}$$

Vemos que o elemento L_{22} (da matriz L), vai diminuindo no decorrer das iterações, até se *anular*. Isto acontece porque na matriz de escalamento, D^*, a segunda e terceira componentes da diagonal, aproximam-se de zero. Isto faz com que a matriz $Q = AD^*A^T$ *perca posto*. Consequentemente, o algoritmo não pode convergir para uma solução primal-dual. Além disto, a cada iteração, a matriz de escalamento D^* vai se tornando muito mal condicionada, pois os autovalores destas matrizes são muito grandes.

Na seção 4.4, veremos que é possível contornar o problema de *perda de posto* através de um processo de *filtragem* da matriz L.

4.2.2 Falta de Pontos Interiores no Problema Dual

O segundo exemplo (exemplo 2) tem três nós e quatro arcos.

Formalmente,

$$\begin{aligned} &Minimizar \quad 3x_1 - 3x_2 + x_3 - x_4 \\ &Sujeito\ a \quad \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \qquad (4.4) \\ &\qquad\qquad (x_1, x_2, x_3, x_4) \geq (0,0,0,0) \end{aligned}$$

Graficamente o problema (4.4) está representado na Figura 4.3.

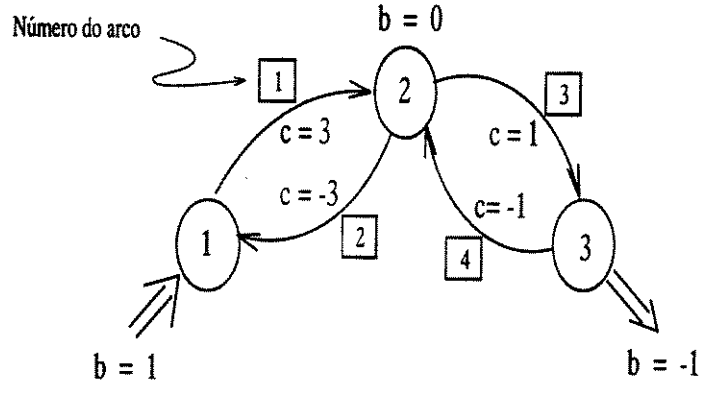


Figura 4.3: Exemplo 2 (3 nós e 4 arcos)

Eliminando-se uma das linhas na matriz de restrições do problema (4.4) (escolheu-

se a última). O dual é:

$$
\begin{array}{lll}
\textit{Maximizar} & y_1 & \\
\textit{Sujeito a} & \begin{pmatrix} 1 & -1 \\ -1 & 1 \\ 0 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} 3 \\ -3 \\ 1 \\ -1 \end{pmatrix} & (4.5) \\
 & (z_1, z_2, z_3, z_4) \geq (0,0,0,0) &
\end{array}
$$

A solução ótima é $x^* = (1+\alpha, \alpha, 1+\beta, \beta)^T \ \forall \alpha, \beta \geq 0$, $y^* = (4,1)^T$ e $z^* = (0,0,0,0)^T$. Nota-se que esta solução é dual degenerada, ou seja, as variáveis de folga são iguais a zero ($z_1^* = z_2^* = z_3^* = 0$). Além disto, o problema (4.5) possui um único ponto extremo. De fato, fazendo-se operações elementares nas restrições do problema (4.5), obtém-se.

$$
\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} z_1 \\ -z_1 \\ z_3 \\ -z_3 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
$$

Logo, $z_1 = 0$, $z_2 = 0$, $z_3 = 0$, e $z_4 = 0$. Usando-se as restrições originais, conclui-se que $y_1 = 4$ e $y_2 = 1$, confirmando a existência de um único ponto extremo.

Vejamos o que acontece com cada um dos métodos de pontos interiores, afins e primais-duais, neste caso.

Solução pelo Método Afim-Primal

Consideremos inicialmente a solução do problema (4.4) pelo método afim-primal. Neste caso, o problema (4.4) tem pontos interiores factíveis, pois o conjunto de soluções factíveis é dado por:

$$
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1+\alpha \\ \alpha \\ 1+\beta \\ \beta \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \forall \alpha, \beta \geq 0
$$

Podemos iniciar o processo de solução com um ponto inicial interior infactível (por exemplo, $x = (1,1,1,1)^T$, tal que $Ax \neq b$). Isto é, podemos tentar obter uma solução inicial com um procedimento semelhante à fase 1 do simplex, discutido no capítulo 2. Vejamos o que acontece com este procedimento (fase 1) quando aplicarmos os passos do método afim-primal. Considere o *problema artificial* a ser resolvido na fase 1:

$$\begin{array}{lll} \textit{Minimizar} & \lambda & \\ \textit{Sujeito a} & \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \lambda\rho = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ & (x_1, x_2, x_3, x_4) \geq (0,0,0,0) \\ & \lambda \geq 0 \end{array}$$

onde $\rho = b - Ax^0$, $x^0 > 0$.

Se encontrarmos uma solução (x^k, λ) tal que $\lambda = 0$, então $x = x^k$ é uma solução interior factível de (4.4).

Vejamos os resultados de cada iteração do processo de solução do problema (4.4) com o procedimento afim-primal. Em cada iteração é mostrada a matriz de escalonamento, D^*, a matriz $Q = AD^*A^T$, o fator de Choleski (matriz L) associado à matriz Q e o vetor de fluxos x.

$$x_{inicial} = (1.000000 \quad 1.000000 \quad 1.000000 \quad 1.000000)^T$$

ITERAÇÃO 1

$$D^* = \begin{pmatrix} 1.000000 & & & \\ & 1.000000 & & \\ & & 1.000000 & \\ & & & 1.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 2.000000 & -1.000000 \\ -1.000000 & 4.000000 \end{pmatrix}$$

$$L = \begin{pmatrix} 1.414214 & 0.000000 \\ -1.414214 & 1.414214 \end{pmatrix}$$

$$x = (1.500000 \quad 0.500000 \quad 1.500000 \quad 0.500000)^T$$

ITERAÇÃO 2

$$
\begin{aligned}
D^* &= \begin{pmatrix} 2.250000 & & & \\ & 0.250000 & & \\ & & 2.250000 & \\ & & & 0.250000 \end{pmatrix} \\
Q &= \begin{pmatrix} 2.500000 & -2.500000 \\ -2.500000 & 5.000000 \end{pmatrix} \\
L &= \begin{pmatrix} 1.581139 & 0.000000 \\ -1.581139 & 1.581139 \end{pmatrix} \\
x &= (1.500000 \quad 0.500000 \quad 1.500000 \quad 0.500000)^T
\end{aligned}
$$

Vemos que o método afim-primal não tem problemas de convergência. Neste caso, a matriz Q mantém-se bem condicionada em todas as iterações. Portanto, se o problema é dual degenerado sem pontos interiores, é recomendável aplicar o método afim-primal.

Solução pelo Método Afim-Dual

A seguir vejamos o que acontece se aplicarmos o método afim-dual para resolver o problema (4.5).

Podemos iniciar o processo de solução com uma aproximação de um ponto inicial interior infactível (um ponto (y, z), tal que $A^T y + z \neq c$, $z > 0$). Isto é, podemos tentar obter uma solução inicial com um procedimento semelhante à fase 1 do simplex, discutido no capítulo 2. Vejamos o que acontece com este procedimento (fase 1) quando aplicarmos os passos do método afim-dual. Considere o *problema artificial* a ser resolvido na fase 1:

$$
\begin{aligned}
&\textit{Minimizar} && \lambda \\
&\textit{Sujeito a} && \begin{pmatrix} 1 & -1 \\ -1 & 1 \\ 0 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} + \lambda\rho = \begin{pmatrix} 3 \\ -3 \\ 1 \\ -1 \end{pmatrix} \\
& && (z_1, z_2, z_3, z_4) \geq (0, 0, 0) \\
& && \lambda \geq 0
\end{aligned}
$$

onde $\rho = c - A^T y^0 - z^0$, $z^0 > 0$.

Se encontrarmos uma solução (y^k, z^k, λ) tal que $\lambda = 0$, então (y^k, z^k) é uma solução interior factível de (4.5).

Vejamos os resultados de cada iteração do processo de solução do problema (4.5) com o procedimento afim-dual. Em cada iteração é mostrada a matriz de escalamento, D^*, a matriz $Q = AD^*A^T$, o fator de Choleski (matriz L) associado à matriz Q e os vetores (y, z).

$$
\begin{aligned}
y_{inicial} &= (0.000000 \quad 0.000000)^T \\
z_{inicial} &= (1.000000 \quad 1.000000 \quad 1.000000 \quad 1.000000)^T
\end{aligned}
$$

ITERAÇÃO 1

$$
D^* = \begin{pmatrix} 1.000000 & & & \\ & 1.000000 & & \\ & & 1.000000 & \\ & & & 1.000000 \end{pmatrix}
$$

$$
Q = \begin{pmatrix} 2.000000 & -2.000000 \\ -2.000000 & 4.000000 \end{pmatrix}
$$

$$
L = \begin{pmatrix} 1.414214 & 0.000000 \\ -1.414214 & 1.414214 \end{pmatrix}
$$

$$
y = \begin{pmatrix} 3.800000 \\ 0.950000 \end{pmatrix} \quad z = \begin{pmatrix} 0.050000 \\ 0.050000 \\ 0.050000 \\ 0.050000 \end{pmatrix}
$$

ITERAÇÃO 2

$$D^* = \begin{pmatrix} 400.000000 & & & \\ & 400.000000 & & \\ & & 400.000000 & \\ & & & 400.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 800.000000 & -800.000000 \\ -800.000000 & 1600.000000 \end{pmatrix}$$

$$L = \begin{pmatrix} 28.284271 & 0.000000 \\ -28.284271 & 28.284271 \end{pmatrix}$$

$$y = \begin{pmatrix} 3.990000 \\ 0.997500 \end{pmatrix} \quad z = \begin{pmatrix} 0.002500 \\ 0.002500 \\ 0.002500 \\ 0.002500 \end{pmatrix}$$

ITERAÇÃO 3

$$D^* = \begin{pmatrix} 160000.000000 & & & \\ & 160000.000000 & & \\ & & 160000.000000 & \\ & & & 160000.000000 \end{pmatrix}$$

$$Q = \begin{pmatrix} 320000.000000 & -320000.000000 \\ -320000.000000 & 640000.000000 \end{pmatrix}$$

$$L = \begin{pmatrix} 565.685425 & 0.000000 \\ -565.685425 & 565.685425 \end{pmatrix}$$

$$y = \begin{pmatrix} 3.999500 \\ 0.999875 \end{pmatrix} \quad z = \begin{pmatrix} 0.000125 \\ 0.000125 \\ 0.000125 \\ 0.000125 \end{pmatrix}$$

ITERAÇÃO 4

$$D^* = \begin{pmatrix} 63999999.998395 & & & \\ & 63999999.999911 & & \\ & & 63999999.998584 & \\ & & & 63999999.999722 \end{pmatrix}$$

$$Q = \begin{pmatrix} 127999999.998306 & -127999999.998306 \\ -127999999.998306 & 255999999.996612 \end{pmatrix}$$

$$L = \begin{pmatrix} 11313.708499 & 0.000000 \\ -11313.708499 & 11313.708499 \end{pmatrix}$$

$$y = \begin{pmatrix} 3.999975 \\ 0.999994 \end{pmatrix} \quad z = \begin{pmatrix} 0.000006 \\ 0.000006 \\ 0.000006 \end{pmatrix}$$

ITERAÇÃO 5

$$D^* = \begin{pmatrix} 25599999999.358154 & & & \\ & 25599999988.448551 & & \\ & & 25599999997.999245 & \\ & & & 25599999989.807461 \end{pmatrix}$$

$$Q = \begin{pmatrix} 51199999987.806702 & -51199999987.806702 \\ -51199999987.806702 & 102399999975.613403 \end{pmatrix}$$

$$L = \begin{pmatrix} 226274.169953 & 0.000000 \\ -226274.169953 & 226274.169953 \end{pmatrix}$$

$$y = \begin{pmatrix} 3.999999 \\ 1.000000 \end{pmatrix} \quad z = \begin{pmatrix} 0.000000 \\ 0.000000 \\ 0.000000 \\ 0.000000 \end{pmatrix}$$

Vemos que o método afim-dual não tem problemas de convergência. Neste caso, a matriz Q mantém-se bem condicionada em todas as iterações.

Solução pelo Método Primal-Dual

Vejamos o que contece se aplicarmos os métodos primais-duais para resolver o problema (4.4) e seu dual (problema 4.5).

Podemos iniciar o processo de solução com um ponto (x, y, z) tal que $x > 0$ e $z > 0$. Ou seja, com um ponto interior primal-dual infactível.

Vejamos os resultados de cada iteração do processo de solução dos problemas (4.4) e (4.5) com o procedimento primal-dual. Em cada iteração é mostrada a matriz de escalamento, D^*, a matriz $Q = AD^*A^T$, o fator de Choleski (matriz L) associado à matriz Q e os vetores (x, y, z).

$$
\begin{aligned}
x_{inicial} &= (1.000000 \quad 1.000000 \quad 1.000000 \quad 1.000000)^T \\
y_{inicial} &= (0.000000 \quad 0.000000)^T \\
z_{inicial} &= (1.000000 \quad 1.000000 \quad 1.000000 \quad 1.000000)^T
\end{aligned}
$$

ITERAÇÃO 1

$$
D^* = \begin{pmatrix} 1.000000 & & & \\ & 1.000000 & & \\ & & 1.000000 & \\ & & & 1.000000 \end{pmatrix}
$$

$$
Q = \begin{pmatrix} 2.000000 & -1.000000 \\ -1.000000 & 4.000000 \end{pmatrix}
$$

$$
L = \begin{pmatrix} 1.414214 & 0.000000 \\ -1.414214 & 1.414214 \end{pmatrix}
$$

$$
x = \begin{pmatrix} 1.500000 \\ 0.500000 \\ 1.500000 \\ 0.500000 \end{pmatrix} \quad y = \begin{pmatrix} 3.331667 \\ 0.999500 \end{pmatrix} \quad z = \begin{pmatrix} 0.000500 \\ 0.666833 \\ 0.000500 \\ 0.666833 \end{pmatrix}
$$

ITERAÇÃO 2

$$D^* = \begin{pmatrix} 3000.000000 & & & \\ & 0.749813 & & \\ & & 3000.000000 & \\ & & & 0.749813 \end{pmatrix}$$

$$Q = \begin{pmatrix} 3000.749813 & -3000.749813 \\ -3000.749813 & 6001.499625 \end{pmatrix}$$

$$L = \begin{pmatrix} 54.779100 & 0.000000 \\ -54.779100 & 54.779100 \end{pmatrix}$$

$$x = \begin{pmatrix} 1.500000 \\ 0.500000 \\ 1.500000 \\ 0.500000 \end{pmatrix} \quad y = \begin{pmatrix} 3.999666 \\ 1.000000 \end{pmatrix} \quad z = \begin{pmatrix} 0.000000 \\ 0.000333 \\ 0.000000 \\ 0.000333 \end{pmatrix}$$

ITERAÇÃO 3

$$D^* = \begin{pmatrix} 6000000.000003 & & & \\ & 1499.625093 & & \\ & & 5999999.995780 & \\ & & & 1499.625093 \end{pmatrix}$$

$$Q = \begin{pmatrix} 6001499.625096 & -6001499.625096 \\ -6001499.625096 & 12002999.245970 \end{pmatrix}$$

$$L = \begin{pmatrix} 2449.795833 & 0.000000 \\ -2449.795833 & 2449.795832 \end{pmatrix}$$

$$x = \begin{pmatrix} 1.500000 \\ 0.500000 \\ 1.500000 \\ 0.500000 \end{pmatrix} \quad y = \begin{pmatrix} 4.000000 \\ 1.000000 \end{pmatrix} \quad z = \begin{pmatrix} 0.000000 \\ 0.000000 \\ 0.000000 \\ 0.000000 \end{pmatrix}$$

Vemos que o método primal-dual não tem problemas de convergência, pois a matriz Q mantém-se bem condicionada em todas as iterações. Além disto, o método converge para a solução ótima somente em 3 iterações.

4.3 Mal Comportados Inofensivos

Nesta seção serão discutidas classes de problemas de fluxo de custo mínimo que, embora apresentem algumas características de mal comportamento, não dificultam a aplicação dos métodos de pontos interiores. No item 1 é apresentado um problema denominado *problema com solução única*, pois possui uma única solução ótima para ambos os problemas primal e dual. A seguir, na seção 2, será apresentado um problema primal, com soluções ótimas degeneradas, mas com pontos interiores factíveis. Finalmente, na seção 3, é feito um estudo de um problema cuja solução ótima é dual degenerada, mas com soluções interiores duais factíveis.

4.3.1 Problema com Solução Única

Consideremos o seguinte problema (exemplo 3). Formalmente,

$$
\begin{aligned}
&\textit{Minimizar} \quad x_1 + 2x_2 + 3x_3 + 4x_4 \\
&\textit{Sujeito a} \quad \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ -2 \\ -1 \end{pmatrix} \qquad (4.6) \\
&\qquad\qquad (x_1, x_2, x_3, x_4) \geq (0,0,0,0)
\end{aligned}
$$

Graficamente o problema (4.6) está representado na Figura 4.4.

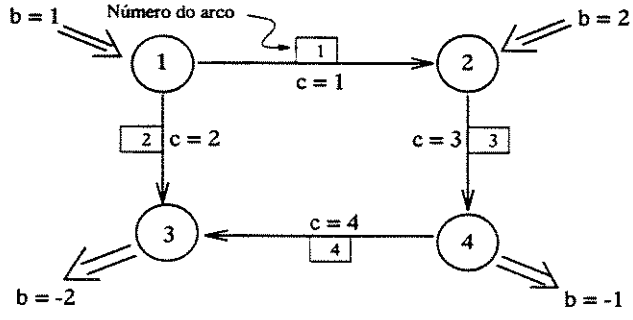


Figura 4.4: Exemplo 3 (4 nós e 4 arcos)

A solução ótima do problema (4.6) é $x^* = (0,1,2,1)^T$.

Eliminando-se uma das linhas na matriz de restrições do problema (4.6) (escolheu-se a última). O dual é:

$$
\begin{array}{ll}
\textit{Maximizar} & y_1 + 2y_2 - 2y_3 \\
\textit{Sujeito a} & \begin{pmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \\
& (z_1, z_2, z_3, z_4) \geq (0,0,0,0)
\end{array} \tag{4.7}
$$

A solução ótima do problema (4.7) é : $y^* = (-2,3,-4)^T$ e $z^* = (6,0,0,0)^T$.

Antes devemos mostrar que ambos problemas (primal e dual) possuem soluções interiores factíveis.

Fazendo pivoteamentos nas restrições do problema (4.6), mostra-se que as soluções interiores primais factíveis estão dadas pela seguinte expressão:

$$
x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \alpha - 1 \\ 2 - \alpha \\ 1 + \alpha \\ \alpha \end{pmatrix} \quad 1 < \alpha < 2
$$

Fazendo pivoteamentos nas restrições do problema (4.7), mostra-se que as soluções interiores duais factíveis estão dadas pela seguinte expressão:

$$
z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} 6 + \alpha - \beta - \gamma \\ \alpha \\ \beta \\ \gamma \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
$$

Além disto, $y = (-2 - \alpha + \gamma, 3 - \beta, -4 + \gamma)^T$.

Neste caso não temos problemas de convergência em nenhum dos métodos de pontos interiores (afins, primais-duais), pois existem soluções interiores factíveis para ambos problemas (primal e dual).

4.3.2 Problema Primal Degenerado com Pontos Interiores

O próximo exemplo (exemplo 4), apresenta um problema cuja solução ótima é primal degenerada, mas com soluções interiores factíveis, permitindo encontrar um ponto inicial interior factível. Para esse problema, pode-se aplicar qualquer dos métodos de pontos interiores (afins ou primais-duais).

Formalmente,

$$
\begin{array}{ll}
\textit{Minimizar} & x_1 + 2x_2 + 3x_3 + 4x_4 \\
\textit{Sujeito a} & \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ -1 \\ -2 \end{pmatrix} \\
 & (x_1, x_2, x_3, x_4) \geq (0,0,0,0)
\end{array} \tag{4.8}
$$

A solução ótima do problema (4.8) é $x^* = (0,1,2,0)^T$. Nota-se que esta solução é primal degenerada, ou seja, existe uma variável básica (x_1 ou x_4) com valor zero.

Graficamente o problema (4.8) está representado na Figura 4.5.

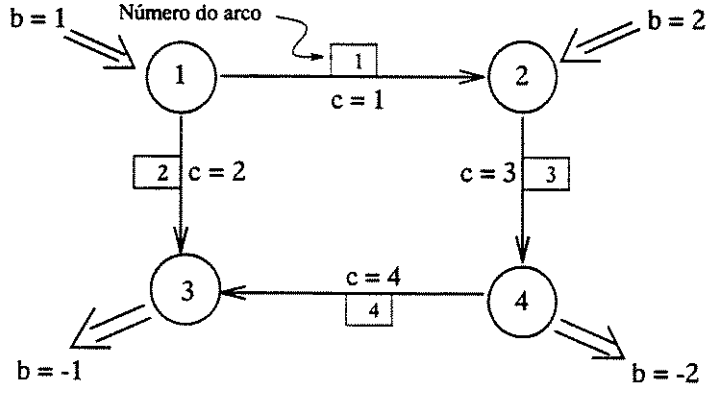


Figura 4.5: Exemplo 4 (4 nós e 4 arcos)

Fazendo pivoteamentos nas restrições do problema (4.8), mostra-se que as soluções

interiores factíveis estão dadas pela seguinte expressão:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \alpha \\ 1-\alpha \\ 2+\alpha \\ \alpha \end{pmatrix}, \quad 0 < \alpha < 1$$

Eliminando-se uma das linhas na matriz de restrições do problema (4.8) (escolheu-se a última). O dual é:

$$\begin{array}{lll} \textit{Maximizar} & y_1 + 2y_2 - y_3 & \\ \textit{Sujeito a} & \begin{pmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} & (4.9) \\ & (z_1, z_2, z_3, z_4) \geq (0,0,0,0) & \end{array}$$

A solução ótima do problema (4.9) é : $y^* = (2+\alpha, 3, \alpha)^T$ e $z^* = (2-\alpha, 0, 0, 4+\alpha)^T$, onde $-4 \leq \alpha \leq 2$.

Neste caso não foram observadas dificuldades de convergência, pois a matriz Q mantém-se bem condicionada em todas as iterações.

4.3.3 Problema Dual Degenerado com Pontos Interiores

Finalmente, é apresentado um problema (exemplo 5), cuja solução ótima é dual degenerada, porém, ele admite soluções interiores duais factíveis, permitindo encontrar um ponto inicial interior dual factível. Assim, pode-se aplicar qualquer dos métodos de pontos interiores para sua solução (afins ou primais-duais).

Formalmente,

$$\begin{array}{lll} \textit{Minimizar} & x_1 + x_2 & \\ \textit{Sujeito a} & \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} & (4.10) \\ & (x_1, x_2) \geq (0,0) & \end{array}$$

Graficamente o problema (4.10) está representado na Figura 4.6.

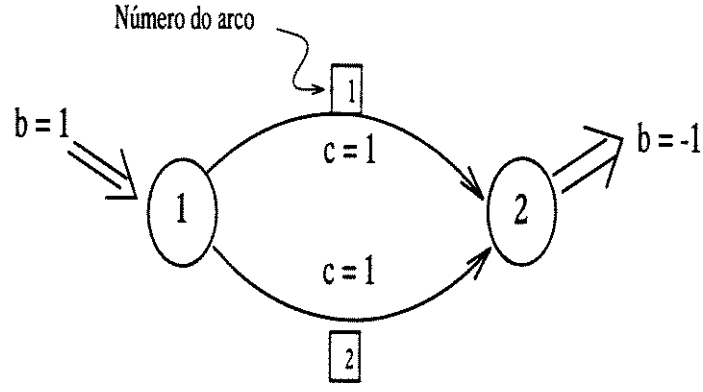


Figura 4.6: Exemplo 5 (2 nós e 2 arcos)

Eliminando-se uma das linhas na matriz de restrições do problema (4.10) (escolheu-se a última), pode-se definir o problema dual.

$$\begin{array}{lll} \textit{Maximizar} & y_1 & \\ \textit{Sujeito a} & \begin{pmatrix} 1 \\ 1 \end{pmatrix} (y_1) + \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} & (4.11) \\ & (z_1, z_2) \geq (0,0) & \end{array}$$

Fazendo pivoteamentos nas restrições do problema (4.11), mostra-se que as soluções interiores duais factíveis estão dadas pela seguinte expressão:

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 1-\gamma \\ 1-\gamma \end{pmatrix} \quad y_1 = \gamma$$

onde $\gamma < 1$.

Neste caso não temos problemas de convergência em nenhum dos métodos de pontos interiores (afins e primais-duais), pois existem pontos interiores duais factíveis.

4.4 Filtragem e Reestruturação do Sistema $(AD^*A^T)y = \hat{b}$

Todos os métodos de pontos interiores resolvem sistemas do tipo $(AD^*A^T)y = \hat{b}$, onde A é a matriz de incidência associada a um problema de fluxo de custo mínimo, D^* é uma matriz de escalamento (diagonal e positiva definida) e os vetores y e $\hat{b}$ representam as direções de deslocamento próprias de cada método. Nos métodos afins e primais-duais, D^* e $\hat{b}$ são dados por:

$$\begin{array}{rll} \text{Afim-primal} & D^* = diag(x_j^2), & \hat{b} \leftarrow AD^*c \\ \text{Afim-dual} & D^* = diag(1/z_j^2), & \hat{b} \leftarrow b \\ \text{Primais-duais} & D^* = diag(x_j/z_j), & \hat{b} \leftarrow b + AZ^{-1}X(c - A^Ty - z) - \mu AZ^{-1}e \end{array}$$

onde $X = diag(x_j)$ e $Z = diag(z_j)$ são matrizes diagonais positiva definidas.

Seja,

$$Q = (AD^*A^T)$$

Para encontrar a solução do sistema $Qy = \hat{b}$, através da fatoração de Choleski, é necessário determinar uma matriz triangular inferior L, com elementos estritamente positivos na diagonal, de tal forma que $LL^T = Q$.

Como foi apresentado na seção 4.2, existem problemas com falta de pontos interiores factíveis. Isto compromete seriamente a convergência dos métodos de pontos interiores. Para contornar esta dificuldade foi aplicado o seguinte procedimento:

Suponha que na $k-$ésima iteração do método de pontos interiores verificou-se que o elemento L_{kk} na diagonal da matriz L apresenta um valor muito próximo de zero ($L_{kk} < \epsilon$, onde ϵ é um parâmetro escolhido de acordo com a precisão da máquina). A filtragem desse elemento consiste basicamente no procedimento descrito a seguir.

Vamos supor que a estrutura corrente da matriz L (calculada em sobreposição à matriz Q) é:

$$L = \begin{pmatrix} L_{11} & & & & & & \\ L_{21} & L_{22} & & & & & \\ \vdots & \vdots & \ddots & & & & \\ L_{j1} & L_{j2} & \cdots & L_{jj} & & & \\ L_{j+1,j} & L_{j+1,2} & \cdots & L_{j+1,j} & \bar{Q}_{j+1,j+1} & \cdots & \bar{Q}_{j+1,m} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ L_{m1} & L_{m2} & \cdots & L_{mj} & \bar{Q}_{m,j+1} & \cdots & \bar{Q}_{mm} \end{pmatrix}$$

onde L_{ij} são os coeficientes do fator de Choleski L já determinados e $\bar{Q}_{ij}$ são os coeficientes da matriz $\bar{Q}$ atualizados.

Suponhamos que $L_{jj} < \epsilon$. Na resolução do sistema $Qy = \hat{b}$, a $j-$ésima linha da matriz L é substituida pelo vetor unitário $e_j = (0, ..., 1, ...0)$ (com valor 1 na $j-$ésima posição) e a $j-$ésima componente do vetor $\hat{b}$ é anulada ($\hat{b}_j = 0$). Isto implica que, na resolução do sistema $Qy = \hat{b}$, tem-se uma linha "quase linearmente dependente" (a $j-$ésima linha) das $(m - j)$ anteriores. Essa linha é eliminada dos cálculos.

O procedimento de filtragem da matriz Q é implementado com o algoritmo a seguir.

$$
\begin{array}{l}
L_{11} = \sqrt{Q_{11}} \\
\texttt{for}\ \ j = 2\ \texttt{to}\ m\ \texttt{do}\ L_{j1} = Q_{j1}/L_{11} \\
\texttt{for}\ \ j = 2\ \texttt{to}\ m\ \texttt{do} \\
\quad S = 0 \\
\quad \texttt{for}\ k = 1\ \texttt{to}\ (j-1)\ \texttt{do}\ S = S + L_{jk}^2 \\
\quad L_{jj} = \sqrt{Q_{jj} - S} \\
\quad \texttt{if}\ L_{jj} > \epsilon \\
\quad \texttt{then} \\
\quad\quad \texttt{for}\ i = j+1\ \texttt{to}\ m\ \texttt{do} \\
\quad\quad\quad S = 0 \\
\quad\quad\quad \texttt{for}\ k = 1\ \texttt{to}\ (j-1)\ \texttt{do}\ S = S + L_{ik} * L_{jk} \\
\quad\quad\quad L_{ij} = (Q_{ij} - S)/L_{jj} \\
\quad \texttt{else} \\
\quad\quad L_{jj} = 1 \\
\quad\quad \texttt{for}\ i = j+1\ \texttt{to}\ m\ \texttt{do}\ L_{ij} = 0
\end{array}
$$

4.5 Recomendações Gerais

Na prática, é muito difícil identificar a priori quando o problema apresenta as características de mal comportamento discutidas neste capítulo. Assim, em geral aplica-se o método (afim-primal, afim-dual ou primal-dual) escolhido para a solução do problema e faz-se a filtragem quando necessário. Se o problema apresentar indícios de falta de pontos interiores e situações semelhantes forem tratadas com frequência, é aconselhável usar o método de solução mais adequado à classe de problemas (como discutido neste capítulo).

Vale lembrar que as situações de falta de pontos interiores ficam mais frequentes em problemas grandes.

Se no problema original tivermos falta de pontos interiores e/ou restrições redundantes, é recomendável aplicar um pré-processamento (na matriz de restrições) para identificá-los a priori. Isto pode evitar algumas das dificuldades discutidas neste capítulo.

Capítulo 5

Análise das Implementações

Neste capítulo são apresentados os resultados computacionais obtidos com a implementação dos diversos métodos de pontos interiores. Os experimentos computacionais foram realizados nas máquinas do Centro Nacional de Processamento de Alto Desempenho em São Paulo (CENAPAD-SP). Utilizou-se uma das estações do cluster IBM Power Risc 6000, com um processador IBM Power Risc 6000 modelo 560. Cada estação do cluster dispõe de 128 Mbytes de memória RAM e 2,5 GB de espaço em disco (o cluster tem ao todo 8 estações).

5.1 Introdução

Considere o par primal-dual de problemas de fluxo de custo mínimo,

$$(P)\ \ Minimizar\ \{\ c^T x\ :\ Ax = b,\ 0 \leq x \leq d\ \}$$

$$(D)\ \ Maximizar\ \{\ b^T y - d^T w\ :\ A^T y - w + z = c,\ \ w, z \geq 0\ \}$$

Para resolver o problema P (e seu dual D), com diferentes características, foram implementados os seguintes métodos de pontos interiores:

- Afins
  - afim-dual.

- Primais Duais com trajetória central
  - primal-dual simples.
  - primal-dual preditor-corretor.

O método afim-primal não foi usado nas comparações, por apresentar dificuldades de convergência em problemas grandes.

Os programas foram batizados com os seguintes nomes:

- **R1** método primal simplex para redes (usado como referência de comparação).
- **R2** método afim-dual com gradiente conjugado pré-condicionado.
- **R3** método primal-dual simples com gradiente conjugado pré-condicionado.
- **R3B** método primal-dual simples com Choleski esparso (grau mínimo).
- **R3C** método primal-dual simples com Choleski esparso (preenchimento local mínimo).
- **R4** método primal-dual preditor-corretor com gradiente conjugado pré-condicionado.
- **R4B** método primal-dual preditor-corretor com Choleski esparso (grau mínimo).
- **R4C** método primal-dual preditor-corretor com Choleski esparso (preenchimento local mínimo).

5.2 Esparsidade da Matriz Q

Diz-se que a matriz Q, $Q = AD^*A^T$, é muito esparsa, se ela possui, no máximo, 5% de elementos não nulos. Definindo-se o grau de esparsidade de Q, $\%NZ(Q)$, como o percentual de elementos não nulos em Q, considera-se os seguintes conjuntos de redes muito esparsas:

(a) Redes com grau de esparsidade menor ou igual a 1% ($\%NZ(Q) \leq 1\%$).

(b) Redes com grau de esparsidade entre 1% e 3% ($1\% < \%NZ(Q) < 3\%$).

(c) Redes com grau de esparsidade entre 3% e 5% ($3\% < \%NZ(Q) < 5\%$).

5.3 Resultados Computacionais

5.3.1 Experimentos com Redes Genéricas

Do grupo (a) foram testadas 5 redes, com número de nós $n = 7000$ e número de arcos de modo atingir o grau de esparsidade desejado. Os resultados computacionais obtidos, são apresentado na tabela 1. Na coluna 1, desta tabela, aparece a dimensão do problema (número de nós - número de arcos), nas colunas 2, 3, 4, 5 e 6, aparecem os métodos R1, R2, R3, R4 e R4B, como definidos acima, respectivamente, e na última coluna (coluna 7) aparece a percentagem de elementos não nulos que tem inicialmente a matriz Q.

TABELA 1

TEMPO DE PROCESSAMENTO EM SEGUNDOS						
Método / Problema	R1	R2	R3	R4	R4B	%NZ(Q)
(7000-7025)	159,66	3536,96	1397,50	1597,58	2166,03	0,16%
(7000-8000)	210,73	3973,50	1437,75	1512,65	7895,16	0,33%
(7000-9000)	298,36	3805,95	1932,71	2094,86	32927,56	0,58%
(7000-9500)	331,00	3832,41	1531,15	1812,48	54980,03	0,75%
(7000-10000)	364,53	3870,23	1452,70	1656,58	84072,71	0,91%
NÚMERO DE ITERAÇÕES						
Método / Problema	R1	R2	R3	R4	R4B	
(7000-7025)	12450	117	38	38	19	
(7000-8000)	16350	127	37	32	18	
(7000-9000)	21968	115	32	30	24	
(7000-9500)	24267	114	35	33	21	
(7000-10000)	27002	113	33	29	21	

Do grupo (b) foram testadas 5 redes, com número de nós $n = 3000$ e número de arcos de modo atingir o grau de esparsidade desejado. Os resultados computacionais obtidos, são apresentados na tabela 2. Na coluna 1, desta tabela, aparece a dimensão do problema (número de nós - número de arcos), nas colunas 2, 3, 4, 5, 6 e 7, aparecem os métodos R1, R2, R3, R3B, R4 e R4B, como definidos acima, respectivamente e na última coluna (coluna 8) aparece a percentagem de elementos não nulos que tem inicialmente a matriz Q.

TABELA 2

TEMPO DE PROCESSAMENTO EM SEGUNDOS							
Método / Problema	R1	R2	R3	R3B	R4	R4B	%NZ(Q)
(3000-4400)	45,03	699,66	254,20	5660,13	279,46	4189,13	1,12%
(3000-4800)	53,93	699,60	300,55	6893,96	340,53	5696,03	1,51%
(3000-5200)	68,21	694,28	270,13	9077,11	321,55	11257,98	1,78%
(3000-5600)	75,20	716,91	276,41	13276,03	328,75	14682,98	2,24%
(3000-6000)	88,26	763,21	301,60	19031,01	388,75	21013,58	2,68%
NÚMERO DE ITERAÇÕES							
Método / Problema	R1	R2	R3	R3B	R4	R4B	
(3000-4400)	10035	114	30	32	24	21	
(3000-4800)	11762	113	35	40	29	18	
(3000-5200)	14332	108	29	35	25	22	
(3000-5600)	15962	108	28	28	23	21	
(3000-6000)	18456	113	31	29	29	19	

Do grupo (c) foram testadas 5 redes, com número de nós $n = 1000$ e número de arcos de modo atingir o grau de esparsidade desejado. Os resultados computacionais obtidos, são apresentado na tabela 3. Na coluna 1 desta tabela aparece a dimensão do problema, nas colunas 2, 3, 4, 5, 6, 7, 8 e 9, aparecem os resultados com os métodos R1, R2, R3, R3B, R3C, R4, R4B e R4C; na última coluna (coluna 10) aparece a percentagem de elementos não nulos que tem inicialmente a matriz Q.

TABELA 3

TEMPO DE PROCESSAMENTO EM SEGUNDOS									
Método / Problema	R1	R2	R3	R3B	R3C	R4	R4B	R4C	%NZ(Q)
(1000-2100)	8,13	89,35	32,93	423,06	1571,96	41,93	298,91	1933,95	3,36%
(1000-2200)	8,95	99,68	34,00	532,95	1912,46	43,75	393,73	2430,20	3,82%
(1000-2300)	9,08	98,78	36,88	623,15	2408,40	48,18	463,30	3060,35	4,11%
(1000-2400)	9,45	106,60	40,36	822,38	2907,58	47,43	616,35	3685,71	4,62%
(1000-2500)	10,85	105,01	41,18	940,81	3398,76	48,11	678,56	4352,98	4,99%
NÚMERO DE ITERAÇÕES									
Método / Problema	R1	R2	R3	R3B	R3C	R4	R4B	R4C	
(1000-2100)	4760	106	25	27	27	21	16	17	
(1000-2200)	5249	115	25	25	25	21	16	17	
(1000-2300)	5276	117	27	26	26	23	16	16	
(1000-2400)	5880	125	30	29	29	24	16	16	
(1000-2500)	6156	118	29	28	28	22	15	15	

Convêm ressaltar que o número de iterações do método simplex não deve ser comparado com o número de iterações dos demais métodos pois tratam-se de iterações bem diferentes.

Considerando os resultados computacionais apresentados nas tabelas 1, 2 e 3, pode-se fazer as seguinte observações:

- Com relação à tabela 1, o método R4B apresenta o menor número de iterações, porém o tempo de processamento é maior do que os outros métodos. Os métodos R1, R3 e R4 são os que apresentam menor tempo de processamento. Os métodos R3B, R3C e R4C não aparecem nesta tabela por apresentar tempos de processamento muito elevados.

- Com relação à tabela 2, novamente, o método R4B apresenta o menor número de iterações, observe-se que este método, faz menos da metade de iterações com relação ao método R3B em um dos problemas testados (problema com 3000 nós e 4800 arcos), mesmo resolvendo dois sistemas a cada iteração. Os métodos R1, R3 e R4 são os que apresentam menor tempo de processamento.

- Com relação à tabela 3, os métodos R3C e R4C são os que apresentam maior tempo de processamento. Os métodos R4B e R4C são os que apresentam menor número de iterações. Novamente, os métodos R1, R3 e R4 são os que apresentam menor tempo de processamento.

Além dos três grupos de redes muito esparsas, foram testadas problemas com graus de esparsidade entre 6% e 22%. Os resultados obtidos são apresentados na tabela 4.

TABELA 4

TEMPO DE PROCESSAMENTO EM SECUNDOS						
Método / Problema	R1	R2	R3	R4	R4B	%NZ(Q)
(1000-3000)	13,28	113,36	44,43	57,58	1766,68	6,82%
(1000-4000)	20,23	129,31	53,45	64,26	3717,03	10,86%
(1000-5000)	27,11	131,73	57,28	74,60	6456,31	14,23%
(1000-6000)	34,15	146,18	64,33	82,43	10185,55	16,88%
(1000-7000)	41,00	157,20	67,43	90,31	13953,13	19,30%
(1000-8000)	48,05	175,33	79,38	105,25	17966,20	21,69%

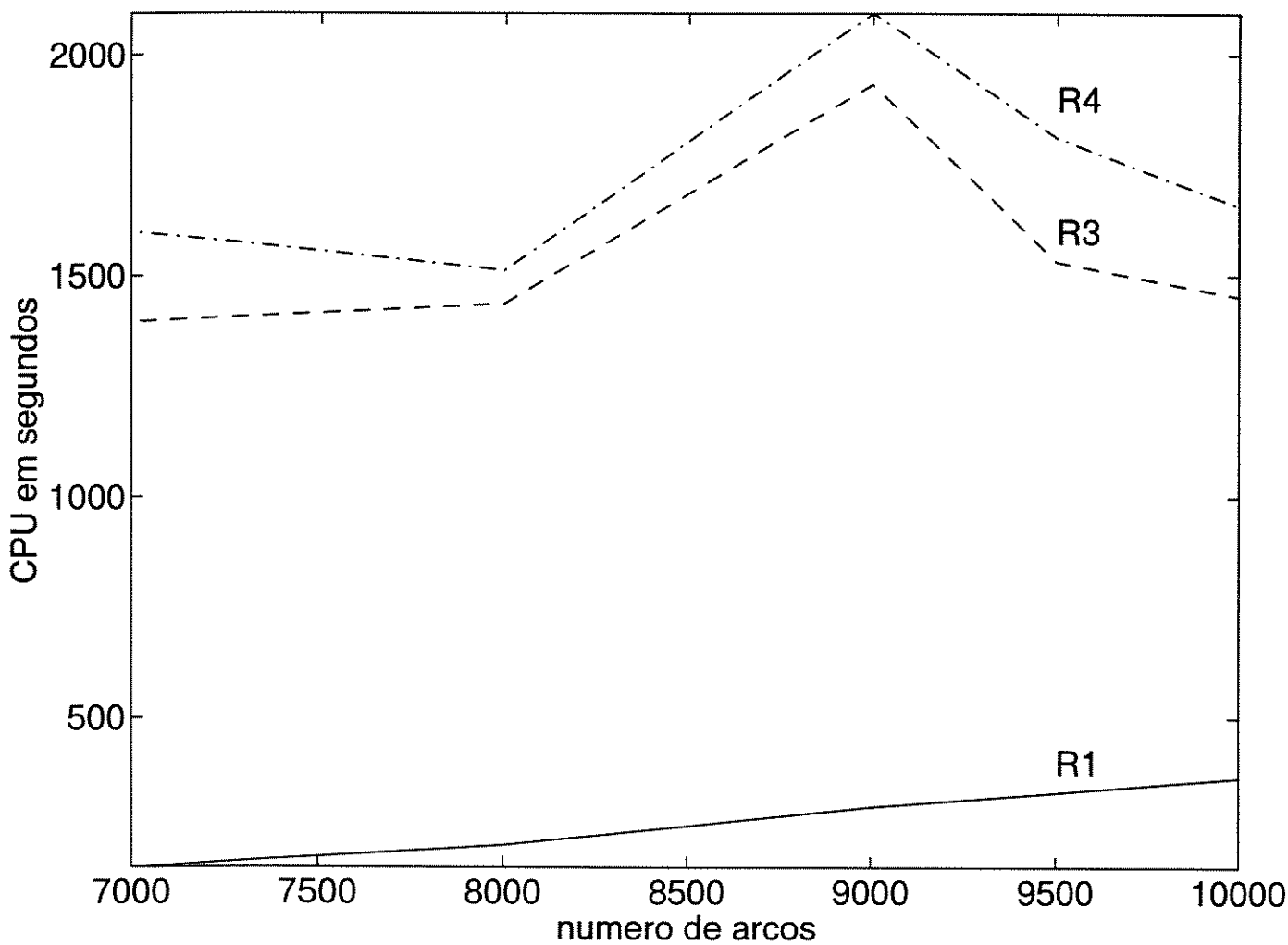
NÚMERO DE ITERAÇÕES					
Método / Problema	R1	R2	R3	R4	R4B
(1000-3000)	8152	117	28	22	17
(1000-4000)	11769	125	29	21	17
(1000-5000)	15648	132	28	21	16
(1000-6000)	19001	138	30	21	19
(1000-7000)	22610	143	29	21	20
(1000-8000)	26412	150	31	22	20

Como podemos verificar a partir da tabela acima, o método R4B (preditor corretor com Choleski esparso) apresenta tempos muito elevados com relação aos outros métodos (R1, R2, R3 e R4). No entanto o número de iterações é menor. Novamente, R1, R3 e R4 são os métodos que apresentam o menor tempo de processamento. Além disto, foi observado que o desempenho do método R3 com relação ao método R1 tende a melhorar a medida que aumenta o número de arcos — se calcularmos o quociente R1/R3 com relação ao primeiro problema (rede de 1000 nós e 3000 arcos) obtém-se $0,29$, enquanto que o valor de R1/R3 associado à rede de 1000 nós e 8000 arcos é $0,60$.

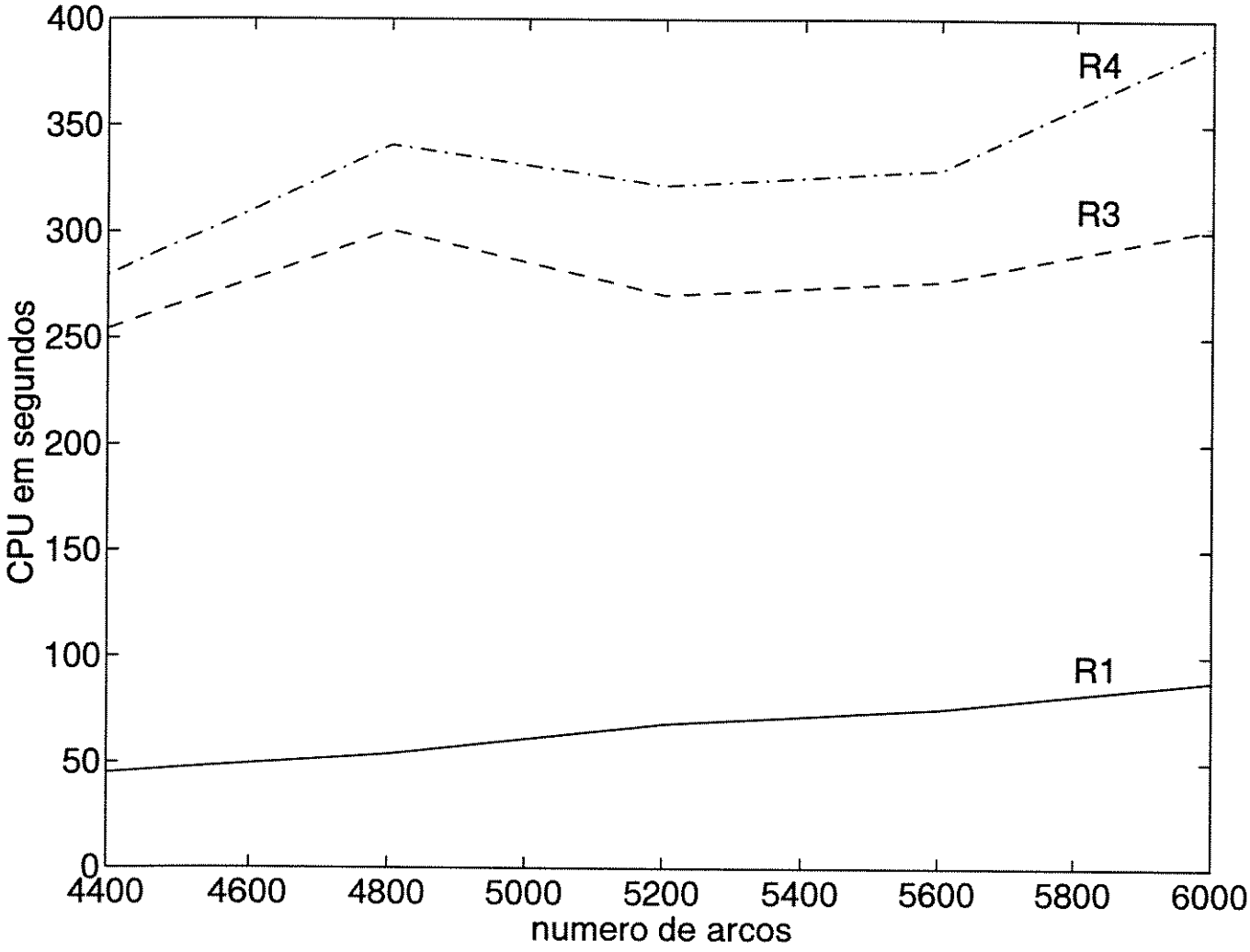
Os gráficos G1, G2, G3 e G4, apresentam o desempenho computacional dos métodos R1, R3 e R4 (os de menor tempo de processamento) nos quatro conjuntos de

resultados das tabelas 1, 2, 3 e 4, respectivamente.

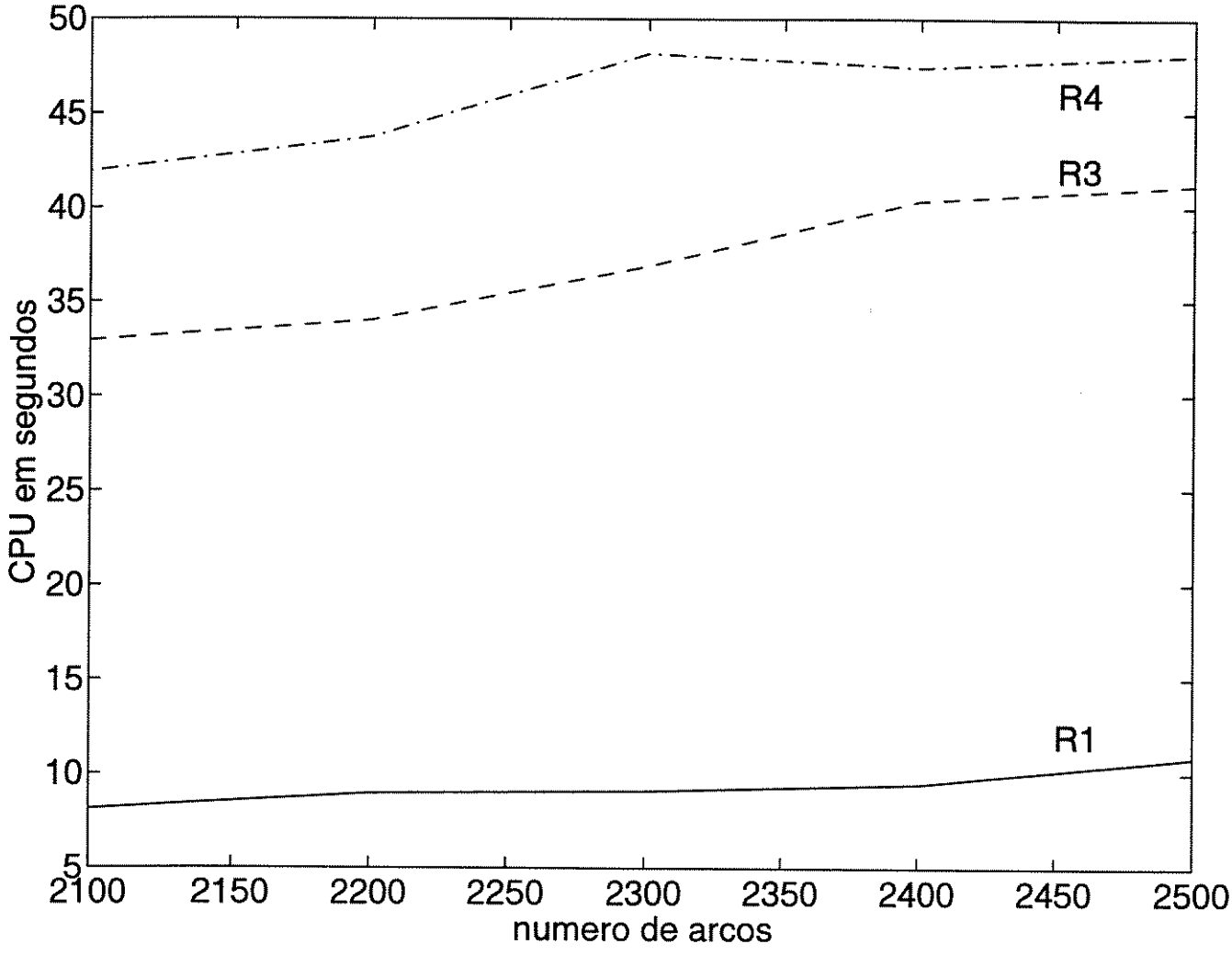
G1: DESEMPENHO DOS MÉTODOS R1, R3 e R4 para $\%NZ(Q) \in (0\%, 1\%)$



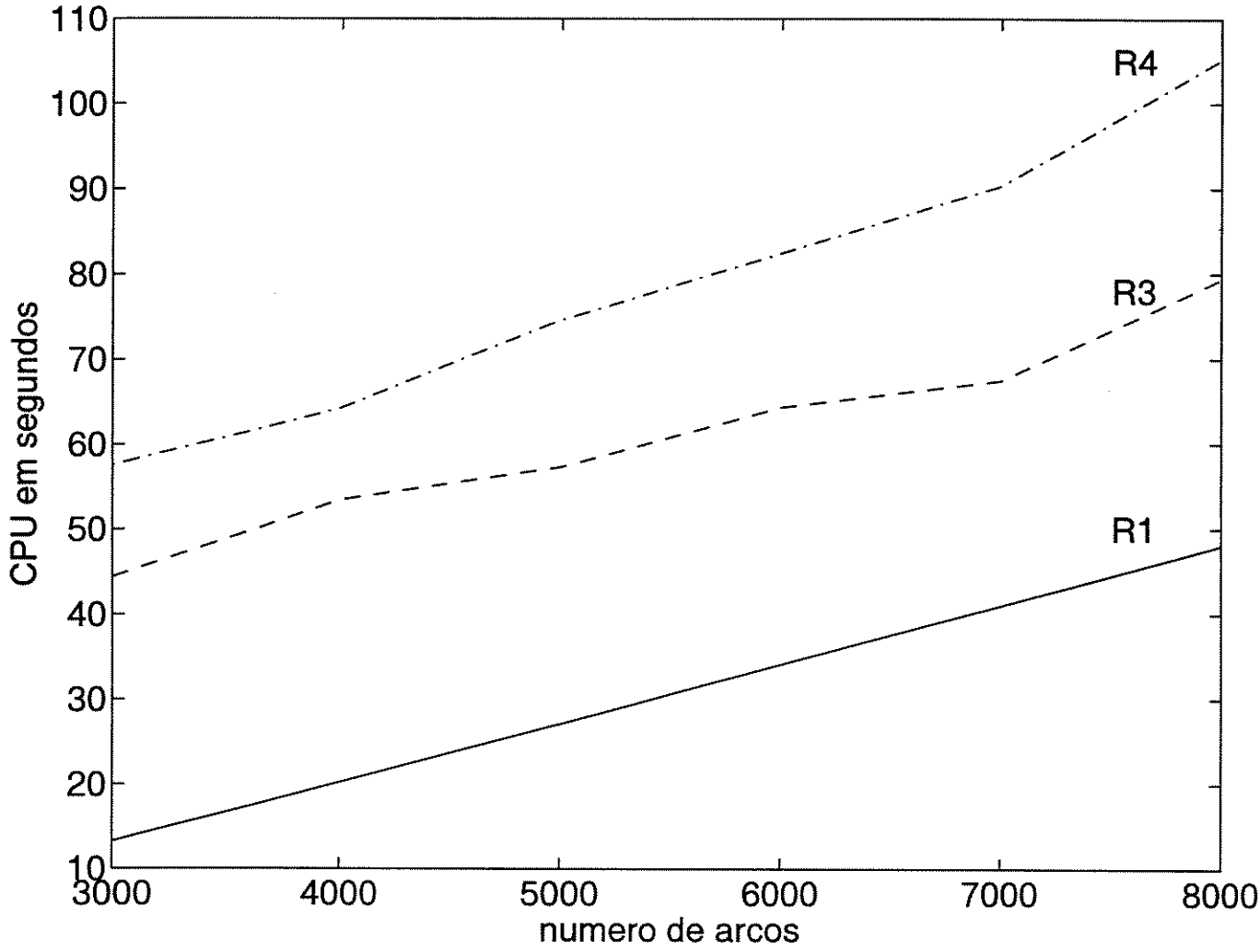
G2: DESEMPENHO DOS MÉTODOS R1, R3 e R4 para $\%NZ(Q) \in (1\%, 3\%)$



G3: DESEMPENHO DOS MÉTODOS R1, R3 e R4 para $\%NZ(Q) \in (3\%, 5\%)$



G4: DESEMPENHO DOS MÉTODOS R1, R3 e R4 para $\%NZ(Q) \in (6\%, 22\%)$



5.3.2 Problemas de Transporte e Designação

No caso do problema de transporte, o grafo associado à matriz de incidência A é um grafo bipartido, isto é, existem dois conjuntos disjuntos de nós (origem e destino) e todos os arcos vão do conjunto de nós origem ao conjunto de nós destino. Neste caso, as matrizes $Q = AD^*A^T$ nos métodos diretos foram reordenadas de modo ótimo, como discutido no capítulo 3.

É possível encontrar uma fórmula fechada para calcular o número de elementos não nulos, $NZ(Q)$, que aparece inicialmente na matriz $Q = AD^*A^T$.

Sejam

r = número de nós origem

s = número de nós destino

O número de elementos não nulos na matriz Q é dado por:

$$NZ(Q) = \begin{cases} (r+s-1)^2 - r(r-1) - (s-1)(s-2) & se \quad r \neq s \\ 2r^2 - 1 & se \quad r = s \end{cases} \tag{5.1}$$

Além disto, como visto no capítulo 3, é possível calcular o número de *fill-ins* que serão gerados após a simulação dos pivoteamentos (fatoração simbólica da matriz Q) quando se utiliza os métodos diretos.

É importante adiantar que, nos problemas de transporte e atribuição, mesmo utilizando ordenação ótima, a matriz Q tem densidade desfavorável para os métodos diretos (fatoração de Choleski); deve-se usar métodos iterativos (gradiente conjugado pré-condicionado) para a resolução do sistema $Qy = \hat{b}$.

Consideremos as seguintes redes:

a) $r = 50$ e $s = 100$, isto é, número de nós origem $<$ número de nós destino. Usando a fórmula (5.1), o número de elementos não nulos que aparece inicialmente na matriz Q é igual a 10049, ou seja, a percentagem de elementos não nulos na matriz Q é igual a $45,26\%$, sem considerar os *fill-ins*.

b) $r = 100$ e $s = 50$, isto é, número de nós origem > número de nós destino. Usando a fórmula (5.1), o número de elementos não nulos que aparece inicialmente na matriz Q é igual a 9949, ou seja, a percentagem de elementos não nulos na matriz Q é igual a $44,81\%$, sem considerar os *fill-ins*.

c) $r = 100$ e $s = 100$, isto é, número de nós origem = número de nós destino. Usando a fórmula (5.1), o número de elementos não nulos que aparece inicialmente na matriz Q é igual a 19999, ou seja, a percentagem de elementos não nulos na matriz Q é igual a $50,50\%$, sem considerar os *fill-ins*.

Os resultados computacionais são mostrados na tabela 5 abaixo. Nesta tabela, na coluna 1 aparece a dimensão do problema (número de nós = $r + s$ e número de arcos), nas coluna 2, 3, 4 e 5, aparecem os resultados da solução dos problemas com os métodos R1, R2, R3 e R4, respectivamente. Na última coluna (coluna 6), aparece a percentagem de elementos não nulos na matriz Q (sem considerar os *fill-ins*).

TABELA 5

TEMPO DE PROCESSAMENTO EM SEGUNDOS					
Método / Problema	R1	R2	R3	R4	%NZ(Q)
(150-4950)	1,55	16,16	16,25	17,96	45,26%
(150-4900)	1,45	25,10	21,35	21,50	44,81%
(200-9900)	4,56	43,51	34,33	45,40	50,50%
NÚMERO DE ITERAÇÕES					
Método / Problema	R1	R2	R3	R4	
(150-4950)	3347	104	34	22	
(150-4900)	3180	101	37	24	
(200-9900)	5690	150	34	22	

O problema de designação é um caso particular do problema de transporte, com um grafo bipartido onde o número de nós origem (r) é igual ao número de nós destino (s), ou seja, a matriz Q é de ordem $(2r-1)$. Usando a fórmula (5.1), conclui-se que inicialmente o número de elementos não nulos na matriz Q é igual a $2r^2-1$, onde r é o número de nós origem (ou nós destino). Para $r = 100$ (100 nós origem e 100 nós destino), a percentagem de elementos não nulos na matriz Q é $50,5\%$, ou seja, aproximadamente metade dos elementos da matriz Q são não nulos (a matriz Q é muito densa).

Mesmo tendo ordenação ótima, Q sofre preenchimento (*fill-ins*). Após a fatoração simbólica (simulação dos pivoteamentos) serão criados $(r-1)(r-2)/2$ elementos novos. No total a matriz Q vai ter $62,75\%$ elementos não nulos, tornando desvantajosa a utilização dos métodos diretos (fatoração de Choleski).

Os resultados computacionais são mostrados na tabela 6.

TABELA 6

TEMPO DE PROCESSAMENTO EM SEGUNDOS					
Método / Problema	R1	R2	R3	R4	%NZ(Q)
(200-10000)	1,53	47,4	27,03	58,01	50,5%
NÚMERO DE ITERAÇÕES					
Método / Problema	R1	R2	R3	R4	
(200-10000)	2996	150	24	24	

5.3.3 Redes com a Mesma Esparsidade e Tamanhos Distintos

Até agora foram apresentados testes computacionais com redes onde o número de nós permaneceu fixo; isto para facilitar o estudo com relação ao grau de esparsidade na matriz Q. Propomos agora avaliar os métodos em problemas onde o número de nós e arcos são aumentados na mesma proporção. Ou seja, procuramos manter o grau de esparsidade na matriz Q aproximadamente igual e verificar o que acontece com o desempenho dos diversos métodos.

Consideremos as seguintes redes onde o número de nós de 5000 nós a 11000 nós (com acréscimos de 1000) e o número de arcos é igual a $1,1$ vezes o número de nós ($n = 1,1m$). Os resultados computacionais obtidos são mostrados na tabela 7.

TABELA 7

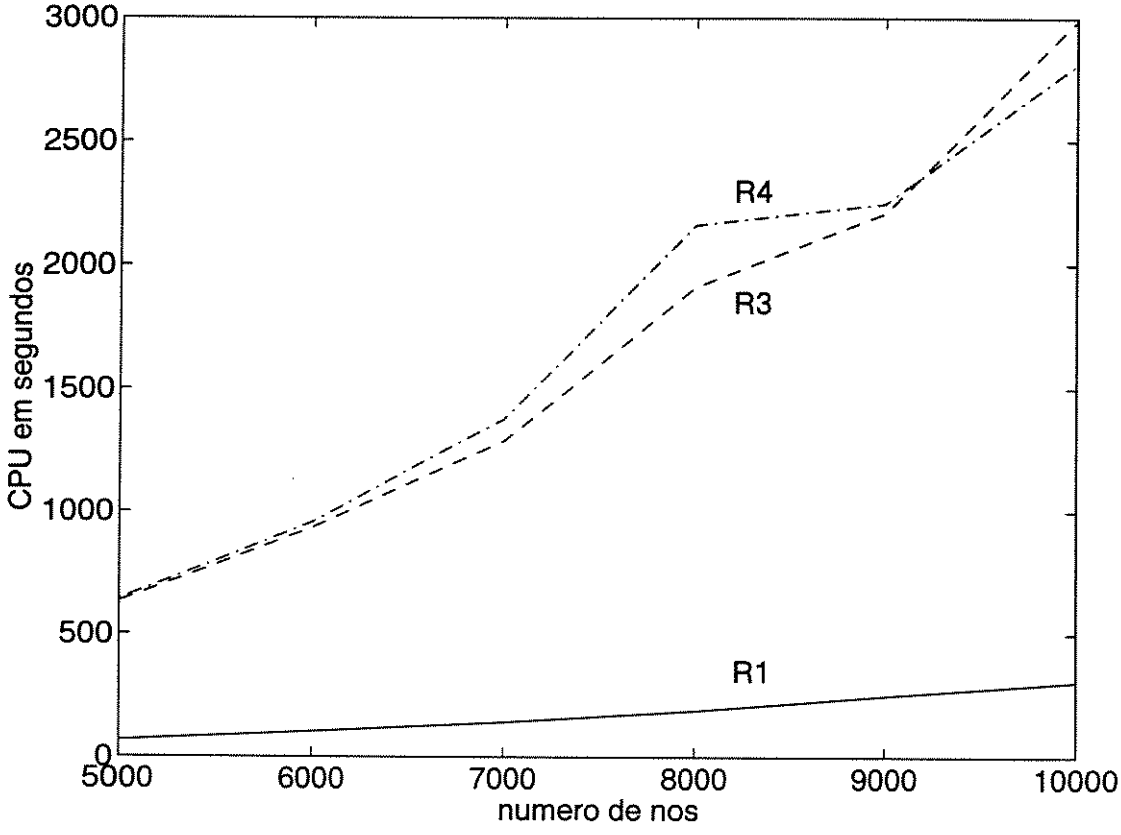
TEMPO DE PROCESSAMENTO EM SEGUNDOS						
Método / Problema	R1	R2	R3	R4	R4B	%NZ(Q)
(5000-5500)	67,98	1678,55	633,41	642,55	797,75	0,269%
(6000-6600)	103,41	2602,36	933,20	955,05	1779,11	0,267%
(7000-7700)	142,38	3602,71	1286,98	1377,23	2860,58	0,257%
(8000-8800)	189,48	4833,01	1912,00	2161,18	5272,31	0,259%
(9000-9900)	250,21	6099,15	2209,75	2248,38	8091,36	0,258%
(10000-11000)	306,61	7672,26	2990,25	2811,33	11678,70	0,249%

NÚMERO DE ITERAÇÕES					
Método / Problema	R1	R2	R3	R4	R4B
(5000-5500)	10218	107	32	26	18
(6000-6600)	12829	115	33	28	23
(7000-7700)	15173	115	33	29	20
(8000-8800)	17521	119	38	37	22
(9000-9900)	20389	119	35	30	19
(10000-11000)	22810	122	39	30	20

Com relação aos resultados obtidos na tabela acima (tabela 7), foi observado o seguinte: os métodos R1, R3 e R4 são os que apresentam o melhor desempenho (menor tempo de processamento). O método R2 apresenta maior tempo de processamento com relação ao método R4B, nos três primeiros problemas testados, revertendo-se esta situação nos outros três problemas. Além disto, o método R4B apresenta o menor número de iterações em todos os problemas testados.

O gráfico G5 mostra o tempo de processamento dos métodos R1, R3 e R4.

G5: DESEMPENHO DOS MÉTODOS R1, R3 e R4



5.3.4 Problemas Muito Grandes

A seguir é apresentado um novo conjunto de testes (6 problemas), onde a característica principal é a existência de um número muito grande de arcos. O que nos levou fazer novos experimentos foi a observação de que na tabela 4 o método R3 (método primal-dual simples com gradiente conjugado pré-condicionado) melhora o desempenho em relação ao método simplex a medida que aumenta o número de arcos.

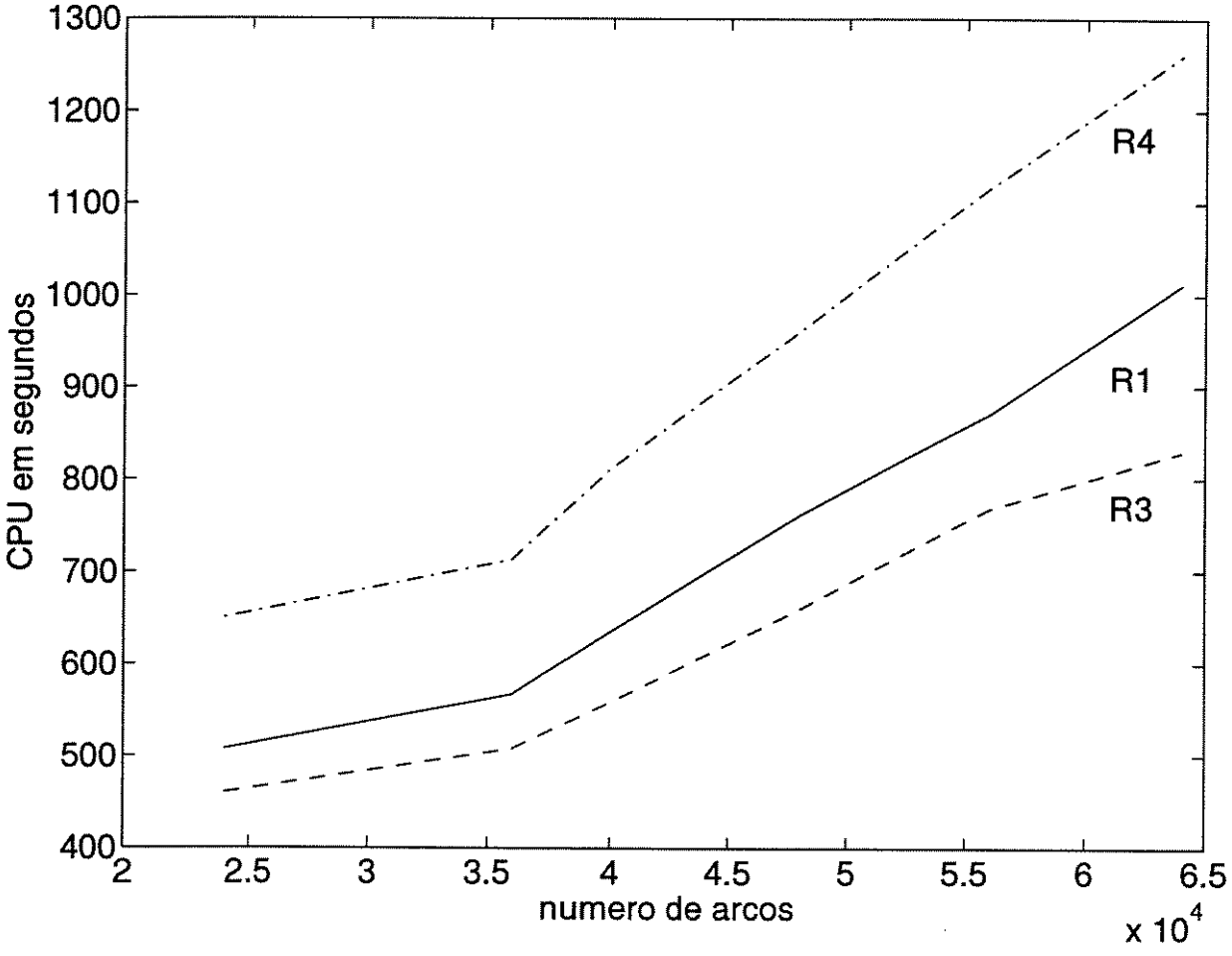
TABELA 8

TEMPO DE PROCESSAMENTO EM SEGUNDOS			
Método / Problema	R1	R3	R4
(2000-24000)	507,76	460,31	650,45
(2000-36000)	567,46	507,96	713,08
(2000-40000)	635,56	559,06	811,73
(2000-48000)	761,53	659,83	961,35
(2000-56000)	870,51	768,01	1117,33
(2000-64000)	1011,65	829,05	1260,11
NÚMERO DE ITERAÇÕES			
Método / Problema	R1	R3	R4
(2000-24000)	138447	34	23
(2000-36000)	153366	36	24
(2000-40000)	172251	37	25
(2000-48000)	207150	39	25
(2000-56000)	234362	39	25
(2000-64000)	272798	38	25

Dos resultados obtidos na tabela 8, conclui-se que se o número de arcos aumenta bastante, o método R3 se torna mais interessante do que o método R1. Além disto, o método R4 apresenta o menor número de iterações. O gráfico G6 ilustra esses resultados.

Como os métodos iterativos não são afetados pela esparsidade da matriz Q, a coluna correspondente a estas informações não foi incluída na tabela 8.

G6: DESEMPENHO DOS MÉTODOS R1, R3 e R4 em PROBLEMAS MUITO GRANDES



5.3.5 Problemas Muito Esparsos

Um último conjunto de testes (10 problemas) dá ênfase à existência de uma quantidade muito pequena de arcos, isto é, a matriz $Q = AD^*A^T$ torna-se muito esparsa. Os resultados computacionais são mostrado na tabela 9. Na primeira coluna desta tabela aparece a dimensão do problema, nas colunas 2, 3, 4, 5, 6, 7, 8 e 9 aparecem os métodos

R1, R2, R3, R3B, R3C, R4, R4B e R4C, respectivamente. Na última coluna (coluna 10) aparece o grau de esparsidade da matriz Q.

TABELA 9

TEMPO DE PROCESSAMENTO EM SEGUNDOS

Método / Problema	R1	R2	R3	R3B	R3C	R4	R4B	R4C	%NZ(Q)
(1000-1005)	1,83	23,48	27,05	9,43	24,63	34,36	6,81	19,78	0,00322
(1000-1010)	1,83	19,96	24,41	9,28	24,51	34,15	6,13	20,83	0,00341
(1000-1020)	1,90	22,60	25,95	9,38	25,25	34,88	6,10	20,70	0,00344
(1000-1030)	1,88	21,68	28,86	10,38	26,40	34,68	6,65	21,56	0,00363
(1000-1040)	1,95	21,16	24,41	9,65	27,21	34,45	7,18	22,91	0,00362
(1000-1050)	2,11	25,10	25,36	10,81	24,90	36,86	6,55	24,40	0,00378
(1000-1060)	2,08	25,60	27,46	11,86	31,41	42,13	8,58	27,21	0,00389
(1000-1070)	2,15	23,31	29,21	12,81	32,73	41,81	8,76	28,50	0,00403
(1000-1080)	2,28	24,76	26,61	13,23	33,35	39,45	9,43	29,35	0,00423
(1000-1090)	2,35	25,23	27,78	13,83	36,03	39,83	9,65	31,68	0,00434

NÚMERO DE ITERAÇÕES

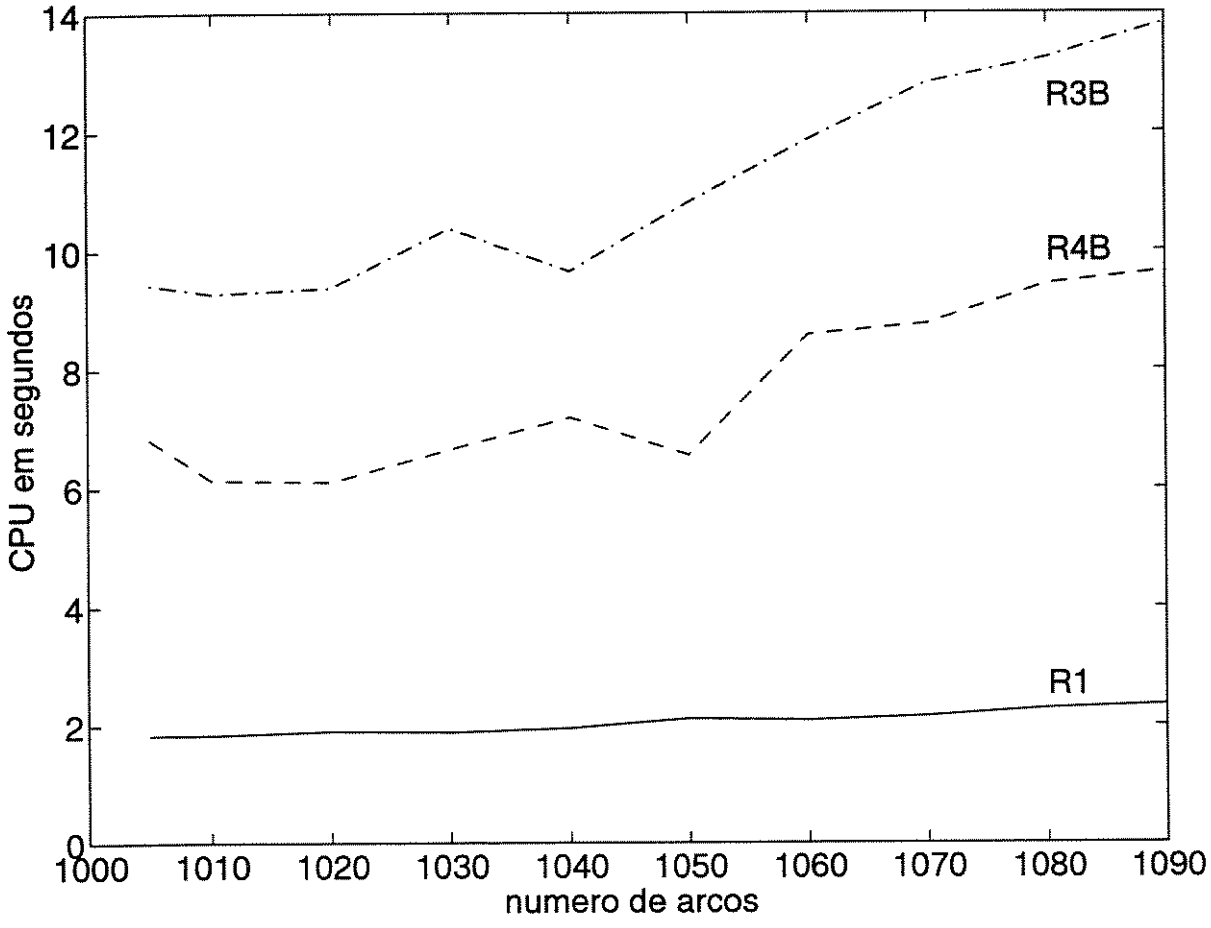
Método / Problema	R1	R2	R3	R3B	R3C	R4	R4B	R4C
(1000-1005)	1455	30	26	24	24	22	20	15
(1000-1010)	1449	25	24	20	20	23	16	17
(1000-1020)	1461	28	24	20	20	21	15	14
(1000-1030)	1475	27	29	22	22	22	16	14
(1000-1040)	1519	26	23	19	19	22	17	14
(1000-1050)	1603	31	23	22	22	22	14	14
(1000-1060)	1596	31	24	24	24	23	19	19
(1000-1070)	1633	28	26	25	25	25	19	19
(1000-1080)	1687	29	23	23	23	22	19	19
(1000-1090)	1721	30	24	24	24	23	19	19

Com relação aos resultados computacionais apresentados na tabela 9, pode-se fazer as seguintes observações: os métodos R1, R3B e R4B apresentam o melhor desempenho, em termos do tempo de processamento. Se compararmos os métodos R3B e R4B (métodos primais duais com Choleski esparso) com relação aos métodos R3 e R4 (métodos primais duais com gradiente conjugado pré-condicionado), vemos que os métodos diretos são melhores. No entanto esta afirmação não permanece verdadeira para redes muito esparsas, porém com maior número de nós (vide tabela 1). Ou seja, se aumenta o número

de nós, mantendo-se constante o grau de esparsidade na matriz Q, os métodos R3 e R4 tornam-se mais eficientes (em termos do tempo de processamento).

O gráfico G7, ilustra o desempenho computacional dos métodos R1, R3B e R4B.

G7: DESEMPENHO DOS MÉTODOS R1, R3B e R4B em REDES MUITO ESPARSAS



5.3.6 Custo da Fatoração Simbólica

Os resultados deste item ilustram a importância do tempo computacional requerido pela fatoração simbólica, usando as heurísticas de ordenação grau mínimo (GM) e preeenchimento local mínimo (PLM), na solução dos problemas apresentados na tabela 9 através dos métodos de pontos interiores R4B e R4C.

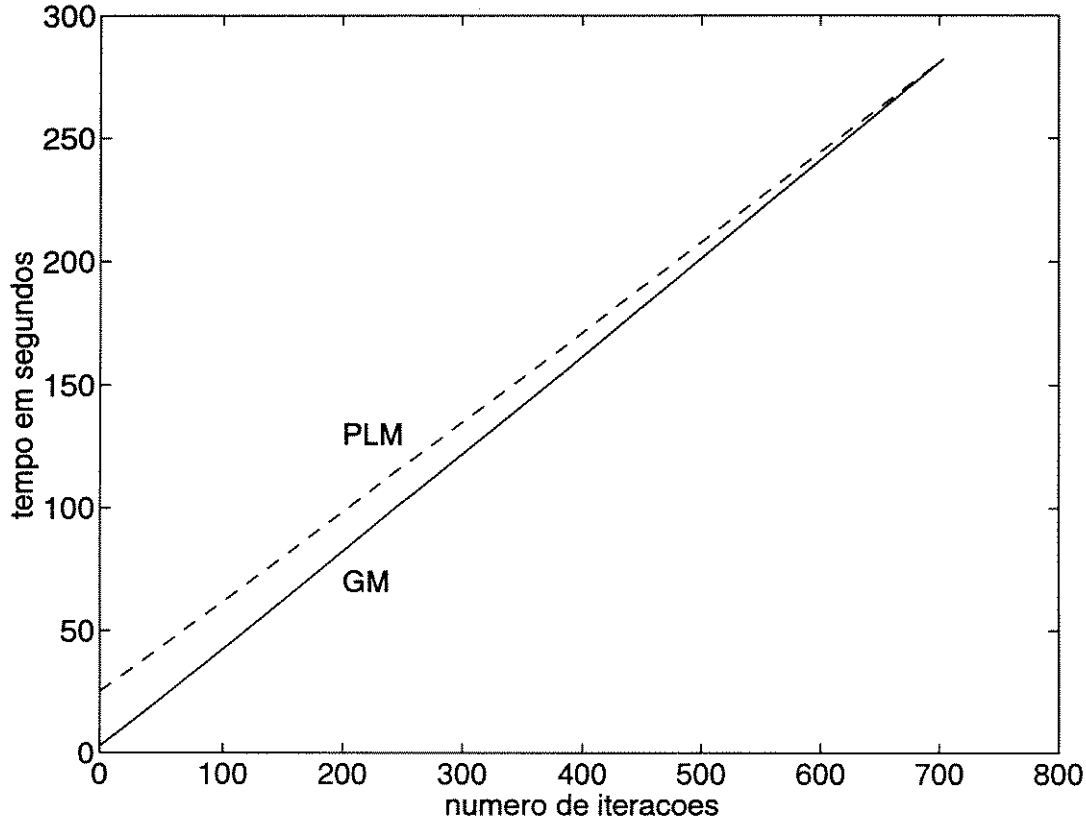
Os resultados computacionais são mostrados na tabela 10. Na primeira coluna desta tabela aparece a dimensão do problema. A coluna 2 mostra a quantidade de elementos não nulos na matriz Q. As colunas 3 e 4 apresentam os elementos novos (preenchimentos), gerados após a fatoração simbólica com as heurísticas GM e PLM, respectivamente. A coluna 5 mostra o tempo de processamento (em segundos) requerido pela heurística GM. A coluna 6 apresenta o tempo médio requerido por uma iteração do método R4B (fatoração numérica e solução triangular). A coluna 7 mostra o tempo de processamento requerido pela heurística PLM. A última coluna (coluna 8) apresenta o tempo médio requerido por uma iteração do método R4C.

TABELA 10

	preenchimento			tempo de processamento (segundos)			
Problema	Inicial	acrescimos		GM		PLM	
		GM	PLM	simb.	1 iter.	simb.	1 iter.
(1000-1005)	3214	1188	1110	1,98	0,252	17,01	0,242
(1000-1010)	3407	1376	1195	1,85	0,267	17,16	0,260
(1000-1020)	3433	1379	1224	2,01	0,272	18,11	0,266
(1000-1030)	3625	1515	1258	2,33	0,297	18,00	0,261
(1000-1040)	3618	1472	1397	2,01	0,288	19,55	0,287
(1000-1050)	3772	1607	1475	2,15	0,304	20,95	0,300
(1000-1060)	3889	1646	1615	2,21	0,323	22,05	0,314
(1000-1070)	4028	1814	1584	2,30	0,342	22,61	0,332
(1000-1080)	4227	1919	1715	2,50	0,382	23,16	0,337
(1000-1090)	4333	1951	1808	2,85	0,397	25,36	0,365

Analizando o problema de 1000 nós e 1090 arcos na tabela 10, vemos que o tempo gasto na fatoração simbólica pelo método GM (2,85 segundos) é bem menor do que o tempo gasto pelo método PLM (25,36 segundos). É possível fazer uma estimativa do número de iterações que seriam necessárias para que o método PLM seja mais interessante (menor tempo de processamento) do que o método GM. O gráfico G8 mostra que seriam necessárias mais de 700 iterações do método de pontos interiores (preditor-corretor) para que o método PLM se torne mais rápido (menor tempo de processamento) do que o método GM. Portanto, não é recomendável usar o método PLM para fazer a fatoração simbólica (simulação dos pivoteamentos).

G8: ESTIMATIVA DO TEMPO COMPUTACIONAL EM FUNÇÃO DO NÚMERO DE ITERAÇÕES



5.4 Conclusões e Informações Complementares

Nesta seção são apresentadas as conclusões decorrentes dos testes computacionais e feitas algumas informações complementares

- Os métodos primal simplex (R1), primal-dual simples com gradiente conjugado pré-condicionado (R3) e preditor-corretor com gradiente conjugado pré-condicionado (R4) são os que apresentam melhor desempenho (menor tempo de processamento), como mostram as tabelas 1, 2, 3, 4 e 7, e os gráficos G1, G2, G3, G4 e G5.
- Em problemas muito esparsos, o método primal simplex (R1) nos parece ainda a alternativa mais eficiente (menor tempo de processamento) para a solução de problemas de fluxo de custo mínimo, como mostram as tabelas 1, 2, 3 e 9, e os gráficos G1, G2, G3 e G7.
- A medida que os problemas aumentam e diminui o grau de esparsidade da matriz Q (o número de arcos da rede aumenta tornando Q menos esparsa), o método primal-dual simples com gradiente conjugado pré-condicionado (R3) vai se tornando mais eficiente (menor tempo de processamento) do que o método primal simplex (R1) — vide tabela 8.
- Apesar da existência de uma ordenação ótima para solução dos problemas de transporte (e designação) pelos métodos diretos, os procedimentos iterativos são mais adequados para estes problemas. Isto é conseqüência da densidade da matriz Q.
- Em relação aos métodos R3 e R4, somente em um dos problemas testados, rede de 10000 nós e 11000 arcos (vide tabela 7), o método preditor-corretor com gradiente conjugado pré-condicionado (R4) mostrou-se mais eficiente (em termos de tempo de processamento) do que o método primal-dual simples com gradiente conjugado pré-condicionado (R3). Seria natural esperar que o método preditor-corretor com gradiente conjugado pré-condicionado (R4) seja, de fato, mais eficiente do que o método primal-dual simples com gradiente conjugado pré-condicionado (R3) para problemas de grande porte, pois trabalha com melhores direções do que os métodos primais-duais (R3). Vale lembrar, no entanto, que apesar das melhores direções do preditor-corretor, ele resolve dois sistemas a cada iteração, sobrecarregando o esforço computacional com os métodos iterativos.

Capítulo 6

Solução de Problemas de Fluxos Generalizados

6.1 Apresentação do Problema

Neste capítulo é apresentado uma extensão dos métodos de pontos interiores para a resolução do problema de fluxo generalizado.

O problema de fluxo generalizado (PFG) é um caso especial de um problema de programação linear, onde cada coluna da matriz de restrições possui no máximo dois elementos não nulos. Formalmente,

$$(PFG)\begin{cases} Minimizar & c^T x & & \\ sujeito\ a & Ax & = & b \\ & x & \geq & 0 \end{cases}$$

onde A é uma matriz de m linhas e n colunas de posto completo (com no máximo dois elementos não nulos em cada coluna). É possível associar um grafo ao problema de fluxo generalizado (PFG). Este grafo consiste de arcos não orientados, como ilustra a Figura 6.1.

Arcos correspondentes a colunas com apenas um elemento não nulo são chamados arcos raiz com o correspondente nó raiz.

Considere o seguinte exemplo (exemplo 1):

$$
\begin{array}{ll}
Minimizar & \left\{ \begin{array}{ccccc} 5 & 10 & 0 & 100 & 100 \end{array} \right\} \; x \\
Sujeito\ a & \left(\begin{array}{ccccc} 4 & & -1 & & \\ 2 & 3 & & 1 & \\ & -4 & -2 & & -1 \end{array} \right) \; x = \left(\begin{array}{c} 0 \\ 4 \\ -8 \end{array} \right) \\
 & x \geq 0
\end{array}
$$

O grafo associado ao exemplo 1 é ilustrado na Figura 6.1.

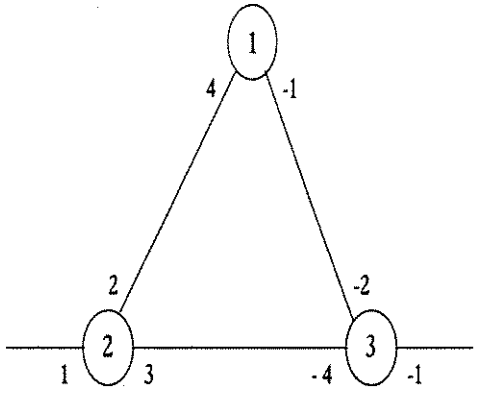


Figura 6.1: Grafo associado ao exemplo 1

6.2 Solução do Sistema $(AD^*A^T)\Delta y = \hat{b}$ Associado ao Problema de Fluxo Generalizado

De forma análoga aos problemas de fluxo de custo mínimo, o maior esforço computacional dos métodos de pontos interiores para a solução do problema de fluxo generalizado (PFG) está na resolução de um sistema simétrico e positivo definido do tipo,

$$AD^*A^T\Delta y = \hat{b} \tag{6.1}$$

onde A é a matriz de restrições associado ao problema de fluxo generalizado (PFG), D^* é uma matriz diagonal (matriz de escalamento), Δy é a direção de deslocamento do método

de pontos interiores e $\hat{b}$ é o lado direito do sistema. No método primal-dual simples, por exemplo, D^* e $\hat{b}$ são dados por:

$$\begin{aligned} D^* &= diag(x_j/z_j) \\ \hat{b} &= b + AD^*(c - A^T y - z) - \mu A Z^{-1} e \end{aligned}$$

O sistema (6.1) pode ser resolvido usando métodos diretos (fatoração de Choleski) ou métodos iterativos (gradiente conjugado pré-condicionado). A seguir é apresentado as especificidades na resolução do sistema (6.1) para cada um desses métodos.

6.2.1 Métodos Diretos: Fatoração de Choleski

Seja,

$$Q = AD^*A^T$$

Para resolver o sistema (6.1), usando a fatoração de Choleski, temos que fazer o seguinte:

(a) Fatoração Simbólica:

Consiste em achar uma matriz de permutação P (ordenação das linhas/colunas na matriz Q), de tal forma que a matriz PQP^T produza uma quantidade razoável de *fill-ins* (poucos *fill-ins*). As heurísticas mais usadas para encontrar esta permutação (matriz P), são *minimum degree* e *minimum local fill-in* (Duff e outros [32]); apresentadas no capítulo 3.

Seja,

$$\bar{Q} = PQP^T$$

(b) Cálculo da matriz $\bar{Q}$:

Uma vez realizada a fatoração simbólica, é preciso calcular a matriz $\bar{Q}$. Para o cálculo desta matriz, deve-se escolher uma estrutura de dados para armazenar os elementos

não nulos da matriz A, os elementos não nulos desta matriz podem ser armazenados em quatro vetores de tamanho n, $t = (t_j)$, $h = (h_j)$, $at = (at_j)$ e $ah = (ah_j)$, onde:

t_j é a cauda do arco j

h_j é a cabeça do arco j

at_j é o coeficiente da cauda do arco j

ah_j é o coeficiente da cabeça do arco j

Sejam,

qij – um nz-vetor que armazena os elementos não nulos que estão fora da diagonal na parte triangular inferior da matriz $\bar{Q}$, por colunas.

qii – um m-vetor que armazena os elementos que estão na diagonal da matriz $\bar{Q}$

ord – um nz-vetor que armazena a ordem na qual os arcos têm que ser considerados para calcular os elementos do vetor qij.

Além disto, define-se um n-vetor $D^* = (d_j)$ que armazena os elementos não nulos da matriz de escalamento.

Para o cálculo da matriz $\bar{Q} = PQP^T$ pode-se adotar o seguinte algoritmo:

For $k = 1$ **to** nz **do** $qij[k] \leftarrow 0$;

For $i = 1$ **to** m **do** $qii[i] \leftarrow 0$;

For $j = 1$ **to** n **do**

$$\begin{array}{lllllll} qij[\ ord_j\ ] & \leftarrow & qij[\ ord_j\ ] & + & at_j * ah_j * d_j \\ qii[\ t_j\ ] & \leftarrow & qii[\ t_j\ ] & + & at_j^2 * d_j \\ qii[\ h_j\ ] & \leftarrow & qii[\ h_j\ ] & + & ah_j^2 * d_j \end{array} \qquad (6.2)$$

O exemplo a seguir ilustra como é obtida a matriz $\bar{Q}$. Usando os dados do

exemplo 1 (vide Figura 6.1), os vetores t, h, at, ah e ord, são dados por:

t	h	at	ah	ord
1	2	4	2	1
2	3	3	−4	3
1	3	−1	−2	2
2	0	1	0	1
3	0	−1	0	2

Além disto,

$$\begin{aligned} D^* &\leftarrow diag(1,1,1,1,1) \quad [matriz\ identidade] \\ qij &\leftarrow (0,0,0) \\ qii &\leftarrow (0,0,0) \end{aligned}$$

Usando o algoritmo apresentado, obtém-se os vetores qij e qii:

$$\begin{aligned} qij &\leftarrow (\ 8 \quad 2 \quad -12) \\ qii &\leftarrow (17 \quad 14 \quad 21) \end{aligned}$$

Logo, tem-se a matriz $\bar{Q}$ abaixo.

$$\bar{Q} = \begin{pmatrix} 17 & & \\ 8 & 14 & \\ 2 & -12 & 21 \end{pmatrix}$$

Observa-se que somente é preciso armazenar a parte triangular inferior da matriz $\bar{Q}$, pois ela é simétrica.

O vetor $\hat{b}$ (lado direito do sistema (6.1)) é expresso na equação abaixo.

$$\hat{b} \ = \ b + AD^*(c - A^T y - z) - \mu AZ^{-1}e$$

$\hat{b}$ pode ser calculado com o algoritmo a seguir.

`For` $i = 1$ `to` m `do` $\hat{b}_i \leftarrow b_i$

`For` $j = 1$ `to` n `do` $cc_j \leftarrow c_j - ah_j * y[h_j] - at_j * y[t_j] - z_j$

For $j = 1$ to n do

$$\hat{b}_i \leftarrow \hat{b}_i + at_j * d_j * cc_j - \mu * at_j / z_j$$
$$\hat{b}_i \leftarrow \hat{b}_i + ah_j * d_j * cc_j - \mu * ah_j / z_j$$

onde o vetor $cc = (cc_j)$ é a infactibilidade do dual, isto é:

$$cc \leftarrow c - A^T y - z$$

Finalmente, devemos calcular o vetor $\hat{b}$ permutado, ou seja, o $m-$vetor $\tilde{b} = P\hat{b}$. Para o cálculo deste vetor usa-se o $m-$vetor $nord = (nord_i)$ que armazena a ordem na qual tem que ser feito os pivoteamentos. Logo, temos:

$$\texttt{For} \quad i = 1 \quad \texttt{to} \quad m \quad \texttt{do} \quad \tilde{b}[nord_i] = \hat{b}_i$$

Voltando à resolução do sistema $(AD^*A^T)\Delta y = \hat{b}$. A solução deste sistema é obtida resolvendo o sistema equivalente:

$$P(AD^*A^T)P^T(P\Delta y) = P\hat{b} \tag{6.3}$$

Seja,

$$\bar{Q} = P(AD^*A^T)P^T = LL^T$$

onde L é o fator de Choleski da matriz $\bar{Q}$.

O sistema (6.3) torna-se:

$$LL^T(P\Delta y) = P\hat{b} \tag{6.4}$$

(c) Fatoração numérica e solução triangular:

Para encontrar a solução do sistema (6.4), faz-se $v = P\Delta y$ e $u = L^T v$. Inicialmente determina-se u tal que $Lu = P\hat{b}$, em seguida resolve-se $L^T v = u$. Finalmente, faz-se $\Delta y = P^T v$.

6.2.2 Métodos Iterativos: Gradiente Conjugado Pré-condicionado

Seja,

$$Q = AD^*A^T$$

O método Gradiente Conjugado é usado para resolver o sistema:

$$(M^{-1}QM^{-1})(M\Delta y) = M^{-1}\hat{b}$$

onde M é uma matriz positiva definida.

O objetivo é fazer a matriz $(M^{-1}QM^{-1})$ menos mal condicionada do que a matriz Q, melhorando a convergência do algoritmo gradiente conjugado. Na Figura 6.2 são apresentados os passos do algoritmo gradiente conjugado pré-condicionado (já apresentado no capítulo 3).

Algoritmo Gradiente-Conjugado-Precondicionado(A,D^*, M, $\hat{b}$, Δy, ϵ)

1	Δy_0	$= 0$		
2	r_0	$= \hat{b}$		
3	Mz_0	$= r_0$		
4	p_0	$= z_0$		
5	i	$= 0$		
6	While	$(\|\|r_i\|\|_2^2$	$>$	$\epsilon)$ Do
	6.1	q_i	$=$	$(AD^*A^T)p_i$
	6.2	α	$=$	$z_i^T r_i / p_i^T q_i$
	6.3	Δy_{i+1}	$=$	$\Delta y_i + \alpha p_i$
	6.4	r_{i+1}	$=$	$r_i - \alpha q_i$
	6.5	Mz_{i+1}	$=$	r_{i+1}
	6.6	β	$=$	$z_{i+1}^T r_{i+1} / z_i^T r_i$
	6.7	p_{i+1}	$=$	$z_{i+1} + \beta p_i$
	6.8	i	$=$	$i + 1$;
7	$\Delta y = \Delta y_i$			

Figura 6.2: Algoritmo Gradiente Conjugado Pré-condicionado

O maior esforço computacional deste algoritmo aparece nas linhas (3, 6.1 e 6.5) (vide figura 6.2). Essas linhas correspondem à multiplicação matriz-vetor (6.1) e à resolução dos sistemas de equações lineares (3 e 6.5). A linha 3 é calculada uma vez, enquanto as linhas (6.1 e 6.5) são calculadas a cada iteração do algoritmo gradiente conjugado. A multiplicação matriz-vetor é da forma AD^*A^Tp e pode ser calculada sem calcular AD^*A^T, explicitamente. Uma maneira muito simples de calcular este produto matriz-vetor é decompor em três multiplicações matriz-vetor.

Fazendo $u = A^Tp$, $v = Du$ e $w = Av$.

Primeiro, calcula-se: $u = A^Tp$

A seguir, determina-se: $v = Du$

Finalmente, calcula-se: $w = Av$

O cálculo dos vetores u, v e w, é realizado da seguinte forma:

$$
\begin{array}{llll}
\texttt{for} \;\; i = 1 \;\; \texttt{to} \;\; m \;\; \texttt{do} \;\; w_i \leftarrow 0 & & & \\
\texttt{for} \;\; j = 1 \;\; \texttt{to} \;\; n \;\; \texttt{do} & & & \\
\quad u_j & \leftarrow & at_j * p[t_j] + ah_j * p[h_j] & [C\acute{a}lculo \;\; de \;\; u] \\
\quad v_j & \leftarrow & d_j * u_j & [C\acute{a}lculo \;\; de \;\; v] \\
\quad w[t_j] & \leftarrow & w[t_j] + at_j * v_j & [C\acute{a}lculo \;\; de \;\; w] \\
\quad w[h_j] & \leftarrow & w[h_j] + ah_j * v_j &
\end{array}
\tag{6.5}
$$

O esforço computacional para calcular u, v e w é $O(n)$, pois no cálculo de u são efetuados n somas, no cálculo de v são efetuados n multiplicações e no cálculo de w são efetuados $2n$ somas. Portanto, em total são efetuados $3n$ somas e n multiplicações.

Usando o algoritmo (6.5) e os dados do exemplo, resumidos na tabela seguir, obtém-se os seguintes valores para os vetores u, v e w.

t	h	at	ah
1	2	4	2
2	3	3	−4
1	3	−1	−2
2	0	1	0
3	0	−1	0

$$\begin{aligned} u &= (8 \quad -6 \quad -7 \quad 2 \quad -3)^T \\ v &= (8 \quad -6 \quad -7 \quad 2 \quad -3)^T \\ w &= (39 \quad 0 \quad 41)^T \end{aligned}$$

Antes da aplicação do algoritmo 6.5, utilizou-se a seguinte inicialização:

$$\begin{aligned} D^* &\leftarrow I \qquad (matriz\ identidade) \\ p &\leftarrow (1 \quad 2 \quad 3)^T \\ w &\leftarrow (0 \quad 0 \quad 0)^T \end{aligned}$$

e $p_0 = w_0 = 0$.

Voltando à resolução dos sistemas (3 e 6.5), no algoritmo gradiente conjugado précondicionado (vide Figura 6.2), deve-se resolver um sistema linear do tipo,

$$Mz = r \tag{6.6}$$

Uma maneira eficiente de resolver este sistema é usando um pré-condicionador (matriz M) que permita encontrar z com pouco esforço computacional e que o número de iterações do gradiente conjugado seja pequeno.

Dois precondicionadores têm-se mostrado eficientes em implementações computacionais para resolver eficientemente o sistema $AD^*A^T\Delta y = \hat{b}$: o pré-condicionador diagonal e o pré-condicionador da árvore geradora máxima.

O precondicionador diagonal, $M = diag(AD^*A^T)$, pode ser calculado em $O(n)$ operações e o sistema (6.6) pode ser resolvido em $O(m)$ divisões.

Também é possível estender para redes generalizadas o pré-condicionador da árvore geradora máxima proposto por Resende e Veiga [88, 89], na implementação do método afim-dual para resolver problemas de fluxo de custo mínimo. Ressalta-se que em geral uma base de um problema de fluxo generalizado corresponde a uma floresta formada por árvores enraizadas ou árvores apoiadas em um (único) ciclo (Kennington e Helgason [58]).

6.3 Métodos de Pontos Interiores para a Resolução do Problema de Fluxo Generalizado

Foram feitas implementações dos métodos de pontos interiores (primais duais e preditor-corretor), apresentados no capítulo 2 (vide seções 2.5.3 e 2.5.4). Além disto, foram implementados as heurísticas de reordenamento das linhas/colunas na matriz $Q = AD^*A^T$, grau mínimo (*minimum degree*) e preenchimento local mínimo (*minimum local fill-in*) para resolver eficientemente o sistema simétrico e definido positivo $Q\Delta y = \hat{b}$.

Os seguintes métodos de pontos interiores foram implementados:

- Método primal–dual simples com Choleski esparso (grau mínimo).
- Método primal–dual simples com Choleski esparso (preenchimento local mínimo).
- Método primal–dual simples com gradiente conjugado pré-condicionado.
- Método preditor–corretor com Choleski esparso (grau mínimo).
- Método preditor–corretor com Choleski esparso (preenchimento local mínimo).
- Método preditor–corretor com gradiente conjugado pré-condicionado.

Todos os programas implementados resolvem o problema de fluxo generalizado com variáveis canalizadas.

Futuramente serão feitos experimentos computacionais comparando o desempenho destes métodos com o método simplex especializado para resolver o problema de fluxo generalizado.

Capítulo 7

Comentários e Conclusões

No presente trabalho foi realizado um estudo cuidadoso dos diversos métodos de solução de problemas de fluxos em redes (problemas de fluxo de custo mínimo e fluxos generalizados), através de métodos de pontos interiores.

No capítulo 2, foram apresentados os principais métodos: afins, afins com centragem e primais-duais. Além disto, foram discutidas inovações em implementações, a partir de interpretações baseadas no método de Newton.

Tendo em vista que a maior parte do esforço computacional dos algoritmos baseados em métodos de pontos interiores é dedicado à solução de sistemas do tipo $Qy = \hat{b}$ (sendo $Q = AD^*A^T$), no capítulo 3 foram estudados diversos métodos para solução desses sistemas (métodos diretos e iterativos) em problemas de fluxo de custo mínimo. Além disto, foi mostrada a identificação de ordenações ótimas para os problemas de atribuição e transportes.

No capítulo 4, foram identificados e analisados problemas com falta de pontos interiores e degenerescências, aspectos frequentes em problemas de fluxos em redes.

No capítulo 5, foram realizadas implementações computacionais e comparações de diversos métodos e alternativas de implementações.

No capítulo 6, foram propostas extensões para a resolução de problemas de fluxos generalizados em redes através de métodos de pontos interiores.

De maneira geral, sobretudo em problemas grandes, os métodos iterativos (gradiente conjugado) com pré-condicionadores bem informados (diagonal e árvore geradora máxima) conseguem aproveitar melhor a estrutura de rede na solução do sistema $AD^*A^Ty = \hat{b}$ — do que, por exemplo, os métodos diretos com fatoração de Choleski (normalmente adotados em problemas lineares genéricos). Isto acontece tanto no problema de fluxos conservativos em redes (PFCM) como no problema de fluxos generalizados.

Na resolução de problemas de programação linear genéricos, o método de pontos interiores preditor-corretor com fatoração de Choleski tem-se mostrado mais eficiente do que o método primal-dual simples com fatoração de Choleski (Lustig, Marsten e Shanno [?]). No entanto, na resolução de problemas de fluxo de custo mínimo, o método prima-dual simples (com gradiente conjugado pré-condicionado) mostrou-se mais eficiente do que o método preditor-corretor (com gradiente conjugado pré-condicionado). Isto acontece porque no método preditor-corretor é necessário resolver dois sistemas (do tipo $AD^*A^Ty = \hat{b}$) a cada iteração do método de pontos interiores, enquanto que no método primal-dual simples é suficiente resolver um sistema do tipo $AD^*A^Ty = \hat{b}$ a cada iteração do método de pontos interiores.

Em linhas gerais, os estudos realizados indicam que os métodos de pontos interiores tornam-se uma alternativa atraente para a resolução de problemas de fluxo de custo mínimo em problemas muito grandes. Para problemas de tamanho médio e problemas pequenos (por exemplo, número de arcos menor do que 20.000), é recomendável usar o método simplex.

Apêndice A

Terminologia de Redes

Seja $G = (\mathcal{N}, \mathcal{A})$ uma rede onde $\mathcal{N} = \{1, ..., n\}$ é o conjunto de nós e $\mathcal{A} = \{1, ..., m\}$ 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.

Uma rede é própria se $n \geq 2$ e $m \geq 1$. Uma rede $G' = (\mathcal{N}', \mathcal{A}')$ é uma sub-rede de G se $\mathcal{N}' \subset \mathcal{N}$ e $\mathcal{A}' \subset \mathcal{A}$. Se $\mathcal{N}' = \mathcal{N}$, então nós dizemos que G' é uma sub-rede geradora de G.

Seja $P = \{s_0, e_1, s_1, e_2, ..., s_{n-1}, e_n, s_n\}$ uma seqüência finita alternada de nós e arcos com $e_j \in \{(s_{i-1}, s_i), (s_i, s_{i-1})\}$. P é um caminho se $s_0, s_1, ..., s_n$ são nós distintos e é um ciclo se $s_1, s_2, ..., s_n$ são nós distintos e $s_0 = s_n$. Portanto, os arcos, tanto de um caminho quanto de um ciclo, são distintos.

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 sub-rede geradora de G é chamada uma árvore geradora de G com nó raiz n, possui $n - 1$ arcos e é denotada por T.

Seja $\hat{A} = (\hat{a}_{ij})$ a matriz de incidência nó-arco associada a G; isto é:

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

Considere a matriz de incidência $\hat{A}$ de uma rede conexa G. Pode-se mostrar que $\hat{A}$ não é de posto completo pois $\sum_i \hat{a}_{ij} = 0, \forall j \in \mathcal{A}$. Além disto, a eliminação de uma linha qualquer de $\hat{A}$ transforma-a numa matriz de posto completo. 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.

Pode-se provar que qualquer conjunto linearmente dependente de colunas de $\hat{A}$ corresponde a uma sub-rede com pelo menos um ciclo e qualquer conjunto gerador de colunas de $\hat{A}$ corresponde a uma sub-rede conexa. Desta forma, 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.

Apêndice B

Simplex em Redes

O algoritmo simplex usado nos testes computacionais, implementado em linguagem C, é essencialmente o método descrito por Kennington e Helgason [58]. Adotou-se a seguinte estrutura de dados:

- $t = (t_j)$ — vetor de n componentes que armazena as caudas dos arcos
- $h = (h_j)$ — vetor de n componentes que armazena as cabeças dos arcos
- $b = (b_i)$ — vetor de m componentes que armazena as demanda nos nós
- $c = (c_j)$ — vetor de n componentes que armazena os custos nos arcos
- $d = (d_j)$ — vetor de n componentes que armazena as capacidades dos arcos
- $x = (x_j)$ — vetor de n componentes que armazena os fluxos nos arcos
- $y = (y_i)$ — vetor de m componentes que armazena os preços nos nós

As bases utilizadas no método simplex são árvores geradoras com raiz fixa no nó raiz. Três vetores (w, u e v) são usados para caracterização das árvores (Kennington e Helgason [58]).

- $w = (w_j)$ — vetor de m componentes que armazena os arcos que antecedem o nó em questão no caminho que parte da raiz até ele.

- $u = (u_i)$ — vetor de m componentes, denominado fio direto, que aponta para o descendente mais próximo de um certo nó, ou para o primeiro descendente do ascendente mais próximo (de um nó folha).
- $v = (v_i)$ — vetor de m componentes, denominado pai, que aponta para o próximo nó no caminho que vai do nó até a raiz.

Seja o nó 1 a raiz e seja a árvore $(\mathcal{N}, T)$ onde $\mathcal{N} = \{1,2,3,4,5\}$ e $T = \{e_1, e_3, e_5.e_7\}$. A figura abaixo ilustra os vetores w, u e v.

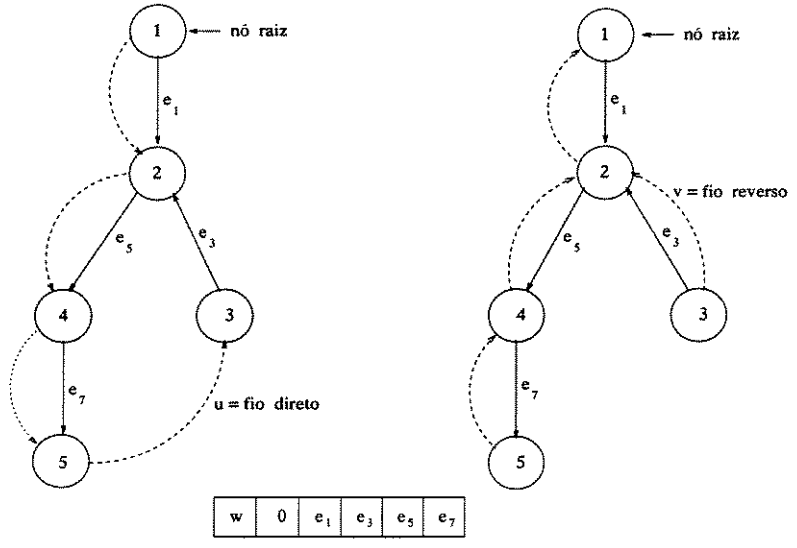


Figura B.1: Vetores w, u e v para percorrer a árvore enraizada T

Apêndice C

Gerador de Redes

Os problemas de fluxo de custo mínimo foram gerados aleatoriamente de acordo com o seguinte esquema:

- m, n, *seed* e *ptrans* fornecidos para cada problema gerado, sendo
  - m — número de nós
  - n — número de arcos
  - *seed* — semente de inicialização do gerador de números aleatórios
  - *ptrans* — probabilidade do nó i ser de transbordo, isto é, $b_i = 0$
- $b_m = -\sum_{i \neq m} b_i$
- c_j — gerado independentemente com distribuição uniforme em (0,10)
- d_j — gerado independentemente com distribuição uniforme em (0,10)

As redes foram geradas de acordo com o seguinte algoritmo:

`Read`(m,n,$ptrans$,$seed$)

```
Begin
    s ⟵ 0
    for i = 2 to m do
        if random(seed) < ptrans then b_i = 0
        else
            b_i ⟵ 20*random(seed) - 10
            s ⟵ s + b_i
    b_1 ⟵ -s
    for j = 1 to n do
        begin
        repeat
            t_j = n*random(seed) + 1
            h_j = n*random(seed) + 1
        until (t_j ≠ h_k) and (b[t_j] ≤ 0) and (b[h_j] ≥ 0)
        c_j ⟵ 10 * random(seed)
        d_j ⟵ 10 * random(seed)
        end
End
```

BIBLIOGRAFIA

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

[2] I. Adler, N. K. Karmarkar, M. G. C. Resende, and G. Veiga, 1989. "Data Structures and Programming Techniques for the Implementation of Karmarkar's Algorithm". *ORSA Journal on Computing* 1, 84-106.

[3] I. Adler, 1989. "A combined 'Phase I-Phase II' Projective Algorithm for Linear Programming". *Mathematical Programming* 43, 209-223.

[4] K. M. Anstreicher, 1986. "A Monotonic Projective Algorithm for Linear Programming". *Algoritmica* 1, 483-498.

[5] K. M. Anstreicher, 1989. "A Combined 'Phase I-Phase II' Projective Algorithm for Linear Programming". *Mathematical Programming* 43, 209-223.

[6] K. M. Anstreicher and R.A. Bosch, 1992. "Long Steps in an O(n L) Algorithm for Linear Programming". *Mathematical Programming* 3, 251-266.

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

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

[9] D. A. Bayer and J.C. Lagarias, 1989. "The Nonlinear Geometry of Linear Programming, Part I. Affine and Projective Scaling Trajectories". *Transactions of the American Mathematical Society* 314, 499-526.

[10] D. A. Bayer and J. C. Lagarias, 1989. "The Nonlinear Geometry of Linear Programming, Part II. Legendre Transform Coordintes and Central Trajectories". *Transactions of the American Mathematical Society* 314, 527-581.

[11] M. S. Bazaraa and J. J. Jarvis, and H. Sherali, 1990. "*Linear Programming and Network Flows*". John Wiley & Sons, New York, NY.

[12] R. C. V. Benito e C. Lyra Filho, 1996. "Solução de Problemas de Fluxos Generalizados Através de Métodos de Pontos Interiores". Anais do VIII CLAIO Latin Iberian-American Congress on Operations Research and System Engineering, pp. 1220-1225, Rio de Janeiro, Agosto de 1996.

[13] R. C. V. Benito e C. Lyra Filho, 1995. "A Matriz AA^T Associada a um Problema de Transportes tem Ordenação Ótima". XXVII SBPO Simpósio Brasileiro de Pesquisa Operacional, Vitória, Novembro de 1995 (*Anais ainda nao publicados*).

[14] R. C. V. Benito e C. Lyra Filho, 1994. "O Método de Newton e os Métodos de Pontos Interiores". Anais do XXVI SBPO Simpósio Brasileiro de Pesquisa Operacional, pp. 653-657, Florianópolis, Novembro de 1994.

[15] R. C. V. Benito e C. Lyra Filho, 1993. "Estudo do Sistema $AA^TX = B$ Associado a um Problema de Fluxos em Redes". Anais do XXV SBPO Simpósio Brasileiro de Pesquisa Operacional, pp. 559-563, Campinas, Novembro de 1993.

[16] D. Bertsekas and D. Castañon, 1989. "The Auction Algorithm for Transportation Problems". *Annals of Operations Research* 20, 67-96.

[17] R. E. Bixby, J. W. Gregory, I. J. Lustig, R. E. Marsten and D. F. Shanno, 1992. "Very Large-Scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods". *Operations Research* 40, 885-897.

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

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

[20] T. J. Carpenter, I. J. Lustig, J. M. Mulvey and D. F. Shanno, 1993. "Higher Order Predictor-Corrector Interior Point Methods with Application to Quadratic Objectives". *SIAM Journal on Optimization* 3, 696-725.

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

[22] V. Chandru and B. S. Kochar, 1985. "A Class of Algorithms for Linear Programming". *Research Memorandum* 85-14, Purdue University.

[23] I. C. Choi, C. L. Monma and D. F. Shanno, 1990. "Futher Development of a Primal-Dual Interior Point Method". *ORSA Journal on Computing* 2, 304-311.

[24] G. B. Dantzig, 1949. "Programming in a Linear Structure". *Econometrica*, Vol. 17, 73-74.

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

[26] G. B. Dantzig, L. R. Ford and D. R. Fulkerson, 1956. "Primal-Dual Alghorithm for Linear Programs". In H. W. Kuhn and A. W. Tucker (eds.), *Linear Inequalities and Related Systems*, Annals of Mathematics Study No. 38, Princeton University Press, Princeton, N.J., 171-181.

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

[28] G. De Ghellinck and J. P. Vial, 1986. "A Polynomial Newton Method for Linear Programming". *Algorithmica* 1:4, 425-453.

[29] D. Den Hertog and C. Roos, 1991. "A Survey of Search Directions in Interior Point Methods for Linear Programming". *Mathematical Programming* 52, 481-509.

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

[31] I. I. Dikin, 1974. "On the Speed of an Iterative Process". *Upravlyaemye Sistemi* 12, 54-60. (in Russian).

[32] I. S. Duff, A. M. Erisman and J. K. Reid, 1986. "*Direct Methods for Sparse Matrices*". Clarendon Press, Oxford, England.

[33] S. C. Fang and S. Puthenpura, 1993. "*Linear Optimization and Extensions: Theory and Algorithms*". Prentice Hall, Engewood Cliffs, New Jersey 07632.

[34] M. Fiedler, 1986. "*Special Matrices and Their Aplications Numeric Mathematics*". Martinus Nijhoff Publishers.

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

[36] D. R. Fulkerson, 1961. "An Out-of-Kilter Method for Minimal-Cost Flow Problems". *J. Soc. Indust. Appl. Math.* 9, 1:18-27.

[37] L. R. Ford and D. R. Fulkerson, 1962. "*Flows in Networks*", Princeton University Press, Princeton, New Jersey.

[38] D. Gale, H. W. Kuhn, and A. W. Tucker, 1951."Linear Programming and the Theory of Games". Chapter 19 of T. C. Koopmans (ed.), *Activity Analysis of Production and Allocation*, Cowles Commission Monograph 13, John Wiley & Sons, New York.

[39] D. M. Gay, 1987. "A Variant of Karmarkar's Linear Programming Algorithm for Problems in Standard Form". *Mathematical Programming* 37, 81-90.

[40] A. George, M. Heath, J. Liu, and E. Ng, 1989. "Solution of Sparse Positive Definite Systems on a Hypercube". *Journal of Computational and Applied Mathematics* 27, 129-156.

[41] A. George and J. W. Liu, 1981. "it Computer Solution of Large Sparse Positive Definite Systems". Prentice Hall, Englewood Cliffs, NJ.

[42] A. George and J. W. Liu, 1989. "The Evolution of the Minimum Degree Ordering Algorithm". *SIAM Review* 31:1, 1-19.

[43] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin and M. H. Wright, 1986. "On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method". *Mathematical Programming* 36, 183-209.

[44] D. Goldfarb and S. Mehrotra, 1988. "A Relaxed Variant of Karmarkar's Algorithm for Linear Programs with Unknown Optimal Objetive Value". *Mathematical Programming* 40, 183-195.

[45] D. Goldfarb and J. K. Reid, 1977. "A Practicable Steepest-Edge Simplex Algorithm", *Mathematical Programming* 12, 361-371.

[46] G. H. Golub and C. F. Van Loan, 1989. "*Matrix Computations*". The Jhons Hopkins University Press.

[47] C. C. Gonzaga, 1989. "An Algorithm for Solving Linear Programming Problems in $O(n^3L)$ operations". *In Nimrod Megiddo, editor. Progress in Mathematical Programming Interior-Point and related methods.* Springer-Verlag, 1-28.

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

[49] C. C. Gonzaga, 1989. "A Conical Projection Algorithms for Linear Programming". *Mathematical Programming* 43, 151-173.

[50] C. C. Gonzaga, 1991. Search Directions for Interior Linear Programmming Methods. *Algorithmica* 2 153-181.

[51] C. C. Gonzaga, 1990/1991. Polinomial Affine Algorithms for Linear Programmming. *Mathematical Programming* 1, 1-722.

[52] C. C. Gonzaga, 1992. Path following methods for Linear Programming. *SIAM Review* 34:2, 167-227.

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

[54] E. L. Johnson, 1965. "Programming in Networks and Graphs". *Technical Report ORC 65 - 1*, Operations Research Center, University of California at Berkeley.

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

[56] N. K. Karmarkar, and K. G. Ramakrishnan, 1991. "Computational Results of an Interior Point Algorithm for Large-Scale Linear Programming". *Mathematical Programming* 52, 555-586.

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

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

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

[60] D. E. Knuth, 1973. "*The Art of Computer Programming: Fundamental Algorithms, Volume I*". Addison Wesley, Reading MA.

[61] M. Kojima, N. Megiddo, and S. Mizuno, 1993. "A Primal-Dual Infeasible-Interior-Point Algorithm for Linear Programming". *Mathematical Programming* 61, 263-280.

[62] M. Kojima, S. Mizuno, and A. Yoshise, 1989. "A Primal-Dual Interior Point Algorithm for Linear Programming". In Nimrod Megiddo, editor. *Progress in mathematical Programming Interior-point and related methods.* Springer-Verlag, New York, 29-47.

[63] C. E. Lemke, 1954. "The Dual Method of Solving the Linear Programming Problem". *Naval Research Logistics Quarterly* 1, 48-54.

[64] D. G. Luenberger, 1984. "*Introduction to Linear and Nonlinear Programming*", Addison-Wesley Publishing Company, Inc.

[65] J. Liu, 1985. "Modification of the Minimum-Degree Algorithm by Multiple Elimination". *ACM Transactions on Mathematical Software* 11, 141-153.

[66] I. J. Lustig, 1991. "Feasibility Issues in a Primal-Dual Interior Point Method for Linear Programming". *Mathematical Programming* 49, 145-162.

[67] I. J. Lustig, R. E. Marsten and D. F. Shanno, 1991. "Computational Experience with a Primal-Dual Interior Point Method for Linear Programming". *Linear Algebra and its Applications* 152, 191-222.

[68] I. J. Lustig, R. E. Marsten and D. F. Shanno, 1991. "Interior Method vs. Simplex Method: Beyond NETLIB, COAL". *Newsletter* 19, 41-44.

[69] I. J. Lustig, R. E. Marsten and D. F. Shanno, 1992. "On Implementing Mehrotra's Preditor-Corretor Interior Point Method for Linear Programming". *SIAM Journal on Optimization* 2, 435-449.

[70] I. J. Lustig, R. E. Marsten and D. F. Shanno, 1994. "Interior Point Methods for Linear Programming: Computational State of the Art". *ORSA Journal on Computing* 1, 1-14.

[71] R. E. Marsten, M. J. Saltzman, D. F. Shanno, J. F. Ballintijn and G.S. Pierce, 1989. "Implementation of a Dual Affine Interior Point Algorithm for Linear Programming". *ORSA Journal on Computing* 1, 287-297.

[72] A. Marxen, 1989. "Primal barrier methods for Linear Programming". *Ph.D. Dissertation.* Department of Operations Research, Stanford University, Princeton University, Princeton, NJ.

[73] K. A. McShane, C. L. Monma, and D. F. Shanno, 1989. "An Implementation of a Primal-Dual Interior Point Method for Linear Programming". *ORSA Journal on Computing* 1, 70-83.

[74] N. Megiddo, 1989. "Pathways to the Optimal Set in Linear Programming", pp. 131-138 in *Progress in Mathematical Programming Interior Point and Related Methods*, N. Megiddo (Ed.), Springer Verlag, NY.

[75] S. Mehrotra, 1991. "Higher Order Methods and their Performance". *Technical Report* 90-16R1. Departament of Industrial Engineering and Management Sciences, Northwestern University (Evanston, IL).

[76] S. Mehrotra, 1991. "On Finding the Optimal Facet of Linear Programs". *Technical Report* 91-10. Departament of Industrial Engineering and Management Sciences, Northwestern University (Evanston, IL).

[77] S. Mehrotra, 1992. "On the Implementation of a Primal-Dual Interior Point Method". *SIAM Journal on Optimization* 2:4, 575-601.

[78] C. Monma and A. J. Morton, 1987. "Computational Experience with a Dual Affine Variant of Karmarkar's Method for Linear Programming". *Bell Communications Research*, Morristown, NJ 07960.

[79] R. D. C. Monteiro, and I. Adler, 1989. "Interior Path Following Primal-Dual Algorithms: Part I: Linear Programming". *Mathematical Programming* 44, 27-41.

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

[81] R. D. C. Monteiro, I. Adler, and M. G. C. Resende, 1990. "A Polynomial-Time Primal-Dual Affine Scaling Algorithm for Linear and Convex Quadratic Programming and its Power Series Extension". *Mathematics of Operations Research.* Vol. 15 No. 2, 191-214.

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

[83] E. C. Ogbuobiri, W. F. Tinney, and J. W. Walker, 1970. "Sparsity Directed Decomposition for Gaussian Elimination on Matrices". *IEEE Trans. Power* PAS-89, 141-150.

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

[85] K. Ramakrishnan, N. Karmarkar, and A. Kamath, 1993. "An Approximate Dual Projective Algorithm for Solving Assignment Problems". In Network Flows and Matching: First DIMACS Implementation Challenge, D. S. Johnson and C. C. McGeoch, eds., vol. 12 of DIMACS Series in Discrete Mathematics and Theorical Computer Science, *American Mathematical Society*, 431-451.

[86] J. Renegar, 1988. "A Polynomial-time Algorithm, Based on Newton's Method, for Linear Programming". *Mathematical Programming* 40, 59-93.

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

[88] M. G. C. Resende and G. Veiga, 1993. "An Implementation of the Dual Affine Scaling Algorithm for Minimum Cost Flow on Bipartite Uncapaciated Networks". *SIAM Journal on Optimization* 3, 516-537.

[89] M. G. C. Resende and G. Veiga, 1993. "An Efficient Implementation of a Network Interior Point Method". In Network Flows and Matching: First DIMACS Implementation Challenge, D. S. Johnson and C. C. McGeoch, eds., vol. 12 of DIMACS Series in Discrete Mathematics and Theorical Computer Science. *American Mathematical Society*, 299-348

[90] D. F. Shanno and A. Bagchi, 1990. "A Unified View of Interior Point Methods for Linear Programming". *Annals of Operations Research* 22, 55-70.

[91] N. Z. Shor and V. I. Gershovich, 1979. "Family of Algorithms for Solving Convex Programming Problems". *Kibernetika* 15, 62-67. (English translation in *Cybernetics* 15(4), 502-507, 1979.)

[92] W. F. Tinney and J. W. Walker, 1967. "Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization". *Proc. IEEE* 55, 1801-1809.

[93] M. J. Todd and B. P. Burrel, 1986. "An Extension of Karmarkar's Algorithm for Linear Programming using Dual Variables". *Algorithmica* 1:4, 409-424.

[94] J. A. Tomlin, 1987. "An Experimental Approach to Karmarkar's Projective Method for Linear Programming". *Mathematical Programming Study* 31, 175-191.

[95] P. Vaidya. "An Algorithm for LP Which Requires $O[((m+n)n+(mn)^{1.5})nL]$ Operations". Presented at the Joint National ORSA/TIMS Meeting, St. Louis, October 1987.

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

[97] R. J. Vanderbei, M. S. Meketon, and B. A. Freedman, 1986. "A Modification of Karmarkar's Linear Programming Algorithm". *Algorithmica* 1:4, 395-407.

[98] M. Yannakakis, 1981. "Computing the Minimum Fill-in is NP-complete". *SIAM J. Alg. Disc. Meth.* 2, 77-79.

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

[100] Y. Ye and M. Kojima, 1987. "Recovering Optimal Dual Solutions in Karmarkar's Polynomial Algorithm for Linear Programming". *Mathematical Programming* 39, 305-317.

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