UNIVERSIDADE ESTATUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

OUTUBRO DE 1989

MÉTODOS DE PONTO INTERIOR EM PROGRAMAÇÃO LINEAR - ESTUDO E IMPLEMENTAÇÃO -

Este exemplar é a versão definitiva da tese e contém as sugestões da banca examinadora.

24/11/89

Aurelio Ribeiro Leite de Oliveira

Orientador *Christiano Lyra Filho*

Tese de Mestrado apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas.

	AMIR	ANA	CELSO	CHRISTIANO	MARCIUS	MARCOS	PAULO
CPFL			X		X	X	X
CTI		X			X		
UNICAMP	X			X			X

Tab. 0 - Participantes

Revisão - Lisete

Tabelas - Luciana

Apresentação

Este trabalho apresenta um estudo das diversas variações de métodos de ponto interior para programação linear surgidas a partir da divulgação do algoritmo polinomial desenvolvido por Karmarkar. Discute também os aspectos mais importantes para uma implementação eficiente destas idéias.

O trabalho original de Karmarkar deixa conceitos como variáveis duais e canalização em aberto e, principalmente, o que diz respeito à implementação computacional do método. Posteriormente, estes pontos foram desenvolvidos independentemente por uma grande quantidade de autores. Parte destas pesquisas é aqui compilada e comentada.

É dada ênfase ao estudo da variante denominada afim, destacando-se diversos pontos relevantes entre os quais variáveis duais, canalizadas, solução inicial e critério de convergência. Esta variante foi implementada em suas formas primal e dual, utilizando técnicas modernas na resolução de sistemas lineares tais como: representação esparsa das matrizes, decomposição, reordenamento da matriz, atualização da decomposição e métodos iterativos.

Uma comparação com o código MINOS para vários problemas mostrou que este método compete favoravelmente com o algoritmo simplex e, nas conclusões, aponta-se tipos de problemas onde pode ser vantajoso utilizar o simplex ou um método de ponto interior.

íNDICE

Capítulo 1

Histórico

1.1 Introdução

As técnicas de otimização vêm sendo cada vez mais utilizadas como ferramentas auxiliares nos processos de tomada de decisões. Estas técnicas buscam a solução de problemas nas mais variadas áreas, através do estabelecimento de um modelo matemático que contempla uma função objetivo e, em geral, um grupo de restrições. As funções objetivo procuram minimizar custos, maximizar lucros, aumentar a eficiência e obter melhores pontos de operação para o sistema, entre várias outras.

A programação linear é a técnica de otimização mais utilizada (uma estatística da IBM de 1970 estima o tempo de utilização de computadores em aplicações científicas da programação linear e técnicas correlatas em 25% do total [47]) devido a robustez e eficiência de seus algoritmos e ao grande número de problemas que podem ser modelados desta forma. Os campos de aplicação desta técnica são tantos que ela chega a ser considerada como um dos mais destacados avanços científicos da segunda metade do século [47]. Como exemplo de áreas de aplicação, podemos citar:

transporte de materiais, atribuição, alocação de recursos, planejamento da operação de sistemas, caminho mínimo, fluxo máximo e otimização de mistura [08,24].

Outro ponto a ressaltar sobre as técnicas de programação linear é a facilidade de se exercer uma análise de sensibilidade na solução ótima, mostrando a variação no comportamento da solução em função de alterações em qualquer parâmetro do problema.

Um problema de programação linear (PL) pode ser estabelecido matematicamente na "forma padrão":

$$
\begin{aligned}
(P) \qquad & \text{Min} \quad cx \\
& \text{s.a.} \quad Ax = b \\
& \qquad\quad x \geq 0
\end{aligned}
$$

Onde A é uma matriz de m linhas por n colunas e os vetores x, c e b têm as dimensões correspondentes.

A forma padrão é a maneira mais usual de apresentar um problema linear, inclusive por ser esta a representação com a qual trabalha o algoritmo simplex. No entanto, alguns dos novos algoritmos de ponto interior trabalham com outras representações.

1.2 Algoritmo Simplex

Desde o início, o método simplex tem sido utilizado como a principal ferramenta de resolução de problemas de programação linear, devido a sua excelente performance e a facilidade de análise de sensibilidade do resultado quando é modificado algum componente do problema.

O conceito teórico por traz do simplex é expresso pelo teorema fundamental da PL: Dado um problema na forma padrão (P), onde A tem posto m, então:

(i) Se existe uma solução factível, então existe um vértice (solução básica) factível;
(ii) Se existe uma solução ótima factível, então existe uma solução básica ótima factível.

O método simplex foi desenvolvido com base no teorema fundamental da PL [24], da seguinte forma: a partir de uma solução básica factível, o método passa para outra solução básica vizinha com valor de função objetivo melhor, até que a solução ótima seja encontrada. Descrições mais completas do método, com discussão de detalhes de implementação, podem ser encontradas em inúmeros bons livros [08,58,72].

Embora o simplex tenha um bom desempenho na média [58,72], ele tem complexidade exponencial, no pior caso, para toda regra de escolha da variável que entra na base conhecida [73,84]. é preciso lembrar que um algoritmo com complexidade exponencial no pior caso não é necessariamente menos eficiente que um algoritmo de complexidade, no pior caso polinomial, em situações que não o pior caso. Em geral, a complexidade no caso médio fornece mais informação, porém é mais difícil de calcular.

O método simplex é citado na literatura [92] como um bom exemplo do cuidado exigido na interpretação da análise de complexidade de algoritmos. Embora ele tenha complexidade exponencial no pior caso, a sua performance média (empírica) é aceita por muitos autores como polinomial [58,72]. Recentemente foi demonstrado que o simplex tem convegência média polinomial com o número de colunas e linhas. Um levantamento destes resultados é feito em [73,92].

A dimensão dos problemas onde se aplica a PL vem crescendo ao longo dos anos, degradando o desempenho médio do simplex. Este fato, aliado a sua complexidade exponencial no pior caso, justifica a procura de um algoritmo de PL que seja polinomial no pior caso e, além desta característica, tenha uma convergência média capaz de competir com o simplex.

1.3 Método dos Elipsóides

O primeiro algoritmo onde se demonstrou que os problemas lineares podem ser resolvidos com complexidade polinomial, no pior caso, foi o método dos elipsóides desenvolvido por Khachiyan [54]. No entanto, este algoritmo teve pouca receptividade por ter obtido resultados piores que o simplex nos testes realizados por vários grupos independentes, além de apresentar problemas de instabilidade numérica [12,73].

O método de Khachiyan, apesar de seus resultados pouco animadores, teve o mérito de despertar a polêmica da existência, ou não, de um algoritmo de resolução de problemas de PL, com complexidade polinomial, que tenha performance na prática competitiva com o método simplex.

1.4 Pontos Interiores

Em 1984, Karmarkar [52] divulgou a existência de um algoritmo, baseado em pontos interiores, de complexidade polinomial, no pior caso, ainda menor que o método dos elipsóides. Na mesma ocasião, mencionou que a implementação por ele utilizada teria uma performance superior ao simplex.

Pouco depois [53], o método foi publicado, embora deixasse muitos pontos em aberto e fizesse algumas hipóteses para o cálculo da complexidade, que dificultavam a implementação. A reserva na apresentação dos resultados teóricos e detalhes de implementação, foi justificada pelo fato do programa ser propriedade da AT&T - mas não só ele, como também a suposta propriedade sobre os teoremas matemáticos. Recentemente, tornou-se corrente que uma das versões deste método já era conhecida desde 1967 [26] e mostrou-se que esta versão pode ser vista como uma especialização de métodos não lineares ("homotopy techniques) para problemas lineares [64].

Várias implementações, feitas a partir das idéias de Karmarkar, não conduziram de imediato a resultados conclusivos sobre a eficiência do algoritmo. No entanto, a discussão sobre a eficácia do método incentivou o surgimento de uma grande quantidade de pesquisas sobre métodos de pontos interiores. Alguns destes métodos serão discutidos no próximo capítulo.

Gill e outros [34] apresentam as primeiras experiências computacionais indicativas de que as idéias de Karmarkar poderiam levar a algoritmos tão (ou mais) eficientes quanto o simplex. Logo em seguida, Adler,

Karmarkar, Resende e Veiga [01] desenvolveram uma variante do método de pontos interiores, que trabalha com a forma dual e usa transformadas afins para reescalar as variáveis do problema. Juntamente com o algoritmo, foram apresentados resultados de testes realizados com uma série de problemas clássicos que permitem comparar o desempenho do método implementado com o código simplex Minos 4.0. Estes resultados confirmaram a competitividade do método que obteve melhor desempenho na maioria dos problemas testados.

Mais recentemente, Vanderbei [80] mostrou que muitos dos métodos, inclusive o tratamento de variáveis canalizadas, poderiam ser obtidos de um algoritmo baseado em uma formulação mais geral de PL. Diante do grande número de variações de métodos de pontos interiores surgidos, foram feitas algumas tentativas de classificá-los, considerando a transformação, função objetivo e cálculo da direção [40,48]. A taxionomia utilizada é a mesma proposta por Hooker [48].

Barnes, Chopra e Jensen [07] propuseram a utilização de centragem como um objetivo de diminuir o número total de iterações dos métodos de ponto interior. A utilização de iterações continuadas proposta aqui também se aplica dentro da linha de tentar obter um menor número de iterações.

Além das diferenças conceituais, existem inúmeras alternativas quanto ao perfil tecnológico da implementação, tais como tratamento da esparsidade, estrutura de dados e decomposição usada na resolução do sistema linear (LU, QR, LL^t, LDL^t), atualização da decomposição, reordenamento, métodos iterativos (gradiente conjugado pré-condicionado), ou ainda o cálculo aproximado da direção. Estes aspectos serão discutidos no capítulo 3 deste trabalho.

Capítulo 2

Métodos de Ponto Interior

2.1 Motivação Geométrica

O método simplex pode ser visto como uma sucessão de passos sobre as arestas do politopo gerado pelo conjunto de restrições. O fato de caminhar pelas arestas determina a sua possível performance exponencial no pior caso devido ao número combinatorial destas. Parece lógico, portanto, que um método eficiente baseado em pontos interiores deva trabalhar tão longe das arestas quanto possível para realizar um passo de bom tamanho (figura 1). A idéia básica dos métodos de ponto interior, surgidos a partir do caminho apontado por Karmarkar [53], consiste em fazer uma transformação que leve o ponto atual ao "centro" (no sentido que todas as variáveis tenham o mesmo valor) do politopo de restrições factíveis. Para realizar esta transformação, é necessário que o ponto seja estritamente interior ao politopo.

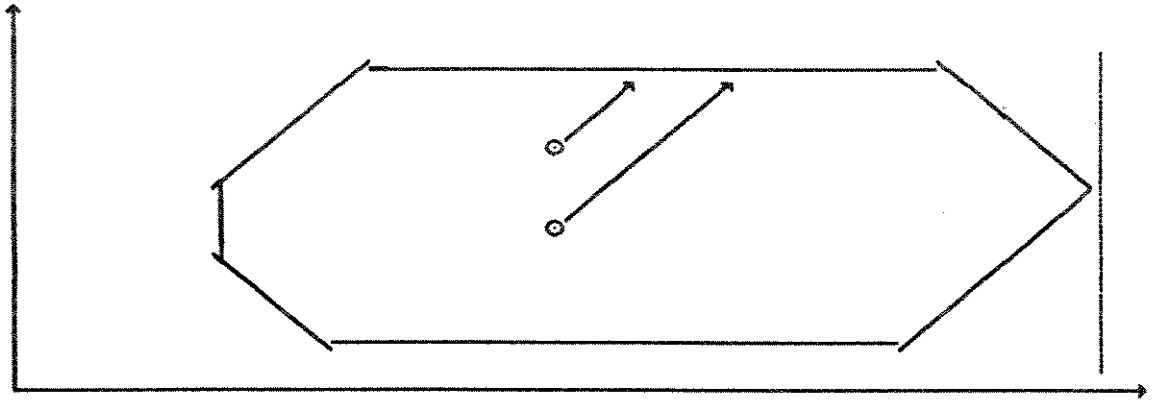


Figura 1.

Uma vez feita a transformação para um ponto central, determina-se uma direção factível (gradiente da função objetivo projetado no espaço nulo da matriz de restrições) de melhora da função, a partir do vetor custo do problema. Encontrada a direção, é feita a transformação inversa para obter a nova solução no problema original.

é necessário que o novo ponto também seja estritamente interior ao politopo para que a próxima iteração seja realizada. Para isto, a maioria dos métodos encontra o tamanho máximo do passo para manter o ponto factível e o encurta em uma pequena percentagem. Na prática, trabalha-se com passos da ordem de 90% do passo máximo. Também é necessário para o cálculo da direção que o posto da matriz de restrições A_{mxn} seja m. Esta condição é suposta verdadeira ao longo de todo este trabalho.

2.2 Método Primal Afim

2.2.1 Desenvolvimento do Algoritmo

Este não foi o método desenvolvido por Karmarkar, no entanto, optamos por apresentá-lo inicialmente por dois motivos: primeiro, por trabalhar com a forma padrão (P) de um problema de PL; segundo, porque a dedução, aqui utilizada, permite uma visão unificadora do método simplex com diversos métodos de ponto interior. O fato deste método trabalhar com a forma padrão também motivou a sua implementação.

Este método foi desenvolvido independentemente por vários autores [06,18,19,26,81]. A dedução vista a seguir foi feita primeiramente por Dikin [26].

O método é fundamentado na minimização dos desvios da condição das folgas complementares. Considere o PL:

$$(P)\ \text{Min } cx \qquad \text{e seu dual } (D)\ \text{Max } by$$
$$\text{sa } Ax = b \qquad\qquad \text{sa } A^t y \leq c$$
$$x \geq 0 \qquad\qquad y \text{ livre}$$

Pelo teorema das folgas complementares, se existirem dois pontos x^* e y^*, respectivamente primal e dual factíveis, tais que $x^*(c - A^t y^*) = 0$, então x^* é ótimo do

primal e y^* é ótimo do dual.

Vamos definir y^k o vetor que minimiza $\gamma^t(y).\gamma(y)$ onde $\gamma(y)$ é o vetor dos desvios da condição de folgas complementares:

$$\gamma(y) = D(c - A^t y) \quad \text{Onde } D = \text{Diag}(x_1, \ldots, x_n)$$

e x um ponto factível do primal.

No ponto ótimo, pelo teorema das folgas complementares,

$$x_i > 0 \longrightarrow A_i^t y = c_i \qquad e \qquad x_i = 0 \longleftarrow A_i^t y < c_i$$

assim, $\gamma(y) = 0$ na otimalidade.

O gradiente de $\gamma^t.\gamma = (Dc - DA^t y)^t(Dc - DA^t y)$ é dado por:

$$2(-DA^t)^t(Dc - DA^t y)$$

No ponto ótimo, $\nabla(\gamma^t\gamma) = 0$; logo:

$$-AD^2c + AD^2A^t y = 0$$

$$y = (AD^2A^t)^{-1}AD^2c$$

A partir da estimativa das variáveis duais, y^k, calcula-se o vetor "custo relativo" $\hat{c} = c - A^t y$ e uma direção descendente factível obtida por um reescalamento (factível) e troca de sinal (descendente) do vetor custo relativo:

$$dx = -D^2\hat{c}$$

A partir da direção calcula-se o novo ponto

de maneira análoga a utilizada pela maioria dos métodos não lineares:

$$x^{k+1} = x^{k} + \alpha dx$$

Esta direção é descendente pois:

$$\gamma^{t}(y).\gamma(y) = (c^{t} - yA)D^{2}(c - A^{t}y) =$$

$$c^{t}D^{2}c - c^{t}D^{2}A^{t}y + y(AD^{2}c - AD^{2}A^{t}y)$$

Substituindo pelo valor de y obtido acima, vem

$$\gamma^{t}(y).\gamma(y) = c^{t}(D^{2}c - D^{2}A^{t}y) = c^{t}D^{2}\hat{c} = -c^{t}dx$$

como $\gamma^{t}(y).\gamma(y) > 0$, $c^{t}dx$ é negativo e:

$$c^{t}(x^{k+1}-x^{k}) = \alpha c^{t}dx < 0 \Longrightarrow c^{t}x^{k+1} < c^{t}x^{k}.$$

é fácil também, verificar que dx é uma direção factível pois:

$$Adx = -AD^{2}c + AD^{2}A^{t}y = -AD^{2}c + AD^{2}c = 0$$

Encontrada a direção, o tamanho máximo do passo é dado pelo primeiro componente de x a se anular, ou seja:

$$\alpha = \beta \text{ Min } (-x_{j}^{k}/dx_{j}^{k} \text{ t.q. } dx_{j}^{k} < 0) \qquad \beta \in (0,1)$$

Dikin [26] mostra que $\alpha = 1 / \|dx\|$ sempre mantém o ponto factível, embora este procedimento em geral obtenha valores muito longe do passo máximo permissível na iteração (nossos experimentos mostraram que isto aumenta, em muito, o número de iterações). Dikin também mostra um

algoritmo baseado nos mesmos princípios discutidos neste item para minimizar funções quadráticas.

A partir dos conceitos acima, pode-se resumir o algoritmo primal afim (A1) apresentado a seguir:

A1.

Seja x^0, um ponto inicial factível,

e $\beta \in (0,1)$

faça $k \leftarrow 0$

Repita

$D \leftarrow Diag(x_1, \ldots, x_n)$ A1.1

$y \leftarrow (AD^2A^t)^{-1}AD^2c$ A1.2

$\hat{c} \leftarrow c - A^t y$ A1.3

$dx \leftarrow -D^2\hat{c}$ A1.4

$\alpha \leftarrow \beta \, Min_i \{ -x_i^k/dx_i^k \text{ t.q. } dx_i^k < 0 \}$ A1.5

$x^{k+1} = x^k + \alpha dx$ A1.6

$k \leftarrow k + 1$

até convergir

Deve-se observar que dois pontos ainda estão em aberto com relação a este algoritmo:

1) Obtenção de um ponto interior inicial;

2) Critério de convergência;

Chandru & Kochar [19] sugerem, como critério de convergência, a verificação das condições das folgas

complementares, ou seja, $\hat{c} \geq 0$ e $\gamma(y) = 0$. Existem, no entanto, outras possibilidades de critério de convergência. Estes critérios serão discutidos em item específico. Note que, como estamos em um ponto estritamente interior, os critérios devem ser atendidos em uma dada tolerância. O aspecto de obtenção de um ponto inicial factível também será discutido em um item específico.

Embora a dedução do algoritmo não tenha utilizado os conceitos geométricos vistos no item anterior, é fácil verificar que a utilização das idéias ali apresentadas leva ao mesmo algoritmo, [81]. Em particular, a direção obtida no algoritmo é a projeção do gradiente da função objetivo do problema transformado no espaço nulo da matriz ($Ad_x = 0$).

2.2.2 Uma Visão do Método Simplex

Chandru & Kochar [19] mostram que este método com uma pequena alteração, quando aplicado a uma solução inicial básica, equivale ao método simplex.

Seja x^k uma solução básica (logo, não interior) e considere a partição I,J entre as variáveis básicas e não básicas, assim $x^k = [x^I, x^J]$ e $A = [A^I | A^J]$

Utilizando também a partição I,J em D, tem-se

$$D = \begin{bmatrix} D^I & 0 \\ 0 & D^J \end{bmatrix} \qquad \text{onde } D^J = 0$$

Calculando as variáveis duais utilizando a fórmula desenvolvida em 2.2.1 (passo A1.2) do algoritmo temos:

$$y = (A^I D_I^2 (A^I)^t + A^J 0 (A^J)^t)^{-1} (A^I D_I^2 c^I + A^J 0 c^J)$$

$$y = (A^I)^{-t} D_I^{-2} (A^I)^{-1} A^I D_I^2 c^I$$

$$\begin{cases} y = c^I (A^I)^{-1} \\ \hat{c}^I = c^I - c^I (A^I)^{-1} A^I = 0 \\ \hat{c}^J = c^J - y A^J \end{cases}$$

A aplicação direta de A1 a uma base fornece uma direção (dx) nula portanto, escolhe-se uma variável x_s com $\hat{c}_s < 0$ para entrar na base, fazendo $D_{ss} = \varepsilon$ (número pequeno positivo) assim, ao aplicarmos o algoritmo, teremos:

$$y = \left[B^{-1} - \frac{\varepsilon^2 B^{-1} A^s (A^s)^t B^{-1}}{1 + \varepsilon^2 (A^s)^t B^{-1} A^s} \right] [A^I D_I^2 c_I + \varepsilon^2 c_s A^s]$$

Onde $B = A^I D_I^2 A^{I^t}$, também foi utilizada a fórmula de atualização de "rank" 1 da inversa (apêndice A).

A partir da fórmula acima, pode-se mostrar que, com algumas simplificações temos:

$$y = (A^I)^{-1} c_I + \varepsilon^2 \theta B^{-1} A^s$$

onde $\theta = \dfrac{\hat{c}_s}{1 + \varepsilon^2 (A^s)^t B^{-1} A^s} = \dfrac{\hat{c}_s}{1 + \varepsilon^2 p p^t} < 0$; pois $\hat{c}_s < 0$.

Ao recalcularmos $\hat{c}$ com o novo valor de y teremos:

$$\hat{c}'_I = -\theta\varepsilon^2 D_I^{-2}(A^I)^{-1}A^s$$

$$\hat{c}'_s = \theta$$

A direção dx fica:

$$dx^I = \varepsilon^2\theta(A^I)^{-1}A^s$$

$$dx^s = -\varepsilon^2\theta$$

$dx^{J-s} = 0$; Pois $D_j^j = 0$ para $j \in J$, $j \neq s$

Note também que novamente $Adx = 0$.

Utilizando $\beta = 1$ para o cálculo do tamanho do passo teremos um procedimento análogo ao pivoteamento no passo A1.6 do algoritmo pois, apenas uma variável não básica se modifica ($d_x^s > 0$) e o tamanho do passo é determinado pela primeira variável básica a se anular (bloqueio) levando a uma nova solução básica. Note que a redefinição de D_s^s altera d_x^I e d_x^s mas mantém as outras componentes nulas. Observe também que a direção calculada não depende de D_s^s. De fato poderíamos escrever $d_x^s = 1$ e $d_x^I = -(A^I)^{-1}A^s$.

Pode-se concluir que tanto o método simplex quanto o primal afim procuram um ponto ótimo através da minimização dos desvios da condição das folgas complementares. O simplex via solução básica; o método primal afim via pontos interiores.

2.3 Método Dual Afim

O algoritmo primal afim tem uma desvantagem do ponto de vista computacional: o cálculo da direção (dx) implica necessariamente em erros de arredondamento, resultando que a relação $Ax^k = b$ não seja estritamente verdadeira. Isto pode causar problemas de convergência.

O método dual afim proposto por Adler Karmarkar, Resende e Veiga, [01] busca resolver este problema. A dedução do método em [01] é diferente da apresentada aqui, novamente baseada na minimização dos desvios das folgas complementares. No entanto, Adler e outros [01] assinalam alguma semelhança no cálculo das variáveis duais em seu trabalho, com o utilizado por Chandru & Kochar [19].

Considere novamente os problemas (P) e (D). Supondo que ambos tenham solução ótima finita, sabe-se que o valor da solução ótima será o mesmo, e que qualquer delas pode ser obtida a partir da outra (x^*, y^*). Portanto, basta resolver um dos problemas para obter a solução de ambos. No item anterior vimos um algoritmo que resolve o problema (P). Veremos agora a versão deste algoritmo que resolve (D).

Primeiramente introduzimos as variáveis de folga v, obtendo uma formulação equivalente para o dual. Assim,

$$
\begin{aligned}
(D) \quad & \text{MAx } by + b'v \\
& \text{sa } A^t y + v = c \\
& v \geq 0 \\
& y \text{ livre}
\end{aligned}
$$

onde $b' = 0$

Pela condição das folgas complementares, no ponto ótimo:

$$x^*(c - A^t y^*) = 0 \qquad \Longrightarrow \qquad x^* v^* = 0$$

Seja $\gamma(x)$ o vetor dos desvios das folgas complementares:

$$\gamma = D^{-1}x \qquad \text{onde} \qquad D = \text{Diag}(1 / v_1, \ldots, 1 / v_n)$$

A definição de D desta forma ($D_i^i = 1 / v_i$) se deve a uma padronização de notação, pois no resultado final teremos novamente um sistema linear com a matriz na forma AD^2A^t (com a definição $D_i^i = v_i$ teríamos $AD^{-2}A^t$).

Analogamente ao primal afim temos:

$$
\begin{aligned}
x = \text{ArgMin } & \gamma^t(x)\gamma(x) \\
\text{sa } & Ax = b
\end{aligned}
$$

ou seja,

$$x^k = \text{ArgMin } |D^{-1}x|^2$$
$$\text{sa} \quad Ax = b$$

O Lagrangeano deste problema é dado por:

$$L(x,\lambda) = (D^{-1}x)^t(D^{-1}x) + \lambda(b - Ax)$$

a solução ótima é dada pelo sitema:

$$\begin{cases} 2D^{-1}(D^{-1}x^k)^t - A^t\lambda = 0 \\ Ax^k = b \end{cases}$$

ou seja,

$$x^k = \frac{1}{2}D^2A^t\lambda$$
$$\lambda = 2(AD^2A^t)^{-1}b$$

logo,

$$x^k = D^2A^t(AD^2A^t)^{-1}b$$

O custo relativo é dado por:

$$\hat{b} = b - Ax = 0$$
$$\hat{b}' = 0 - Ix = -x$$

Reescalando $\hat{b}'$ temos:

$$dv = -D^{-2}x$$

Uma vez que $v = c - A^ty$ temos também que calcular a direção das variáveis duais y de forma a manter a factibilidade, assim:

$$v + \alpha dv = c - A^t(y + \alpha dy)$$

$$dv = -A^t dy$$

$$-D^{-2}x = -A^t dy$$

$$x = D^2 A^t dy$$

Substituindo x por seu valor obtido acima temos:

$$D^2 A^t (AD^2 A^t)^{-1} b = D^2 A^t dy$$

$$AD^2 A^t (AD^2 A^t)^{-1} b = AD^2 A^t dy$$

$$b = AD^2 A^t dy$$

$$dy = (AD^2 A^t)^{-1} b$$

Pode-se verificar que a direção dy é ascendente, pois:

$$\gamma^t \gamma = x^t D^{-2} x = b^t (AD^2 A^t)^{-1} AD^2 D^{-2} D^2 A^t (AD^2 A^t)^{-1} b$$

$$\gamma^t \gamma = b^t (AD^2 A^t)^{-1} b = b^t dy > 0$$

uma vez que $b^t dy$ é positivo podemos concluir que:

$$b^t y^{k+1} > b^t y^k$$

Neste caso, sendo as variáveis y irrestritas, o tamanho máximo do passo é dado pela primeira variável de folga a se anular:

$$\alpha = \beta \text{ Min } \{ -v_j^k / dv_j^k \text{ t.q. } dv_j^k < 0 \} \qquad \beta \in (0,1)$$

O algoritmo dual afim (A2) pode ser resumido da seguinte forma:

seja y^0 um ponto inicial factível e $\beta \in (0,1)$

$k \leftarrow 0$

A2.

Repita

$v \leftarrow c - A^t y$ A2.1

$D \leftarrow Diag(1/v_1, \ldots, 1/v_n)$ A2.2

$dy \leftarrow (AD^2A^t)^{-1}b$ A2.3

[$x \leftarrow D^2 A^t dy$] A2.4

$dv \leftarrow -A^t dy$ A2.5

$\alpha \leftarrow \beta$ Min ($-v_j^k/dv_j^k$ t.q. $dv_j^k < 0$) A2.6

$y^{k+1} = y^k + \alpha dy$ A2.7

$k \leftarrow k + 1$

até convergir

Neste algoritmo, o cálculo das variáveis primais (A2.4), entre colchetes, não é necessário a cada iteração, exceto se o valor da função objetivo primal for utilizado como critério de convergência.

O algoritmo é de fácil implementação e numa comparação com o código Minos 4.0, para a resolução de vários problemas clássicos, obteve uma performance superior [01].

Observe que a transformação afim nas variáveis de folga ($D^{-1}v$) leva o ponto atual ao centro do politopo do problema dual (no sentido que a distância do ponto a qualquer restrição é igual), pois as variáveis duais y são irrestritas. Assim como no algoritmo primal afim, a dedução poderia ter sido feita pela visualização geométrica do método. Neste caso, aplicada a restrições de desigualdade e variáveis irrestritas (que não necessitam ser reescaladas).

Cavalier & Schall [17] adaptaram o método primal afim para o seguinte problema:

$$\begin{aligned} \text{Max } & cx \\ \text{sa } & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Esta forma difere do problema dual (D) somente pelas restrições $x \geq 0$ determinando um algoritmo diferente. A vantagem deste método em relação ao primal afim está justamente nas restrições de desigualdade que evitam eventuais problemas numéricos, pois as variáveis de folga podem ser calculadas em função do ponto atual a cada iteração, como no algoritmo dual afim.

Uma observação interessante é que o método dual afim resolve o problema dual (D), ao contrário do

método dual simplex que trabalha no problema primal (P). De fato, como será visto mais adiante, o algoritmo dual afim é o próprio algoritmo primal afim aplicado a variáveis irrestritas e restrições do tipo $Ax \leq b$.

Um algoritmo primal-dual é proposto por Megiddo [61] e estudado por Kojima e outros [57]. O algoritmo se baseia na aplicação do método das barreiras com função logarítmica. Ele necessita de uma solução inicial primal e dual factível (x^0, y^0). Observa-se no entanto que o método de inicialização proposto invibializa a utilização do algoritmo em problemas esparsos, pois acrescenta uma coluna densa na matriz A, tornando a matriz AD^2A^t completamente densa. Métodos de inicialização para os algoritmos primal afim e dual afim que evitam este problema serão vistos no item sobre inicialização. Monteiro e Adler [90] modificaram o tamanho do passo neste algoritmo e obtiveram uma convergência melhor no pior caso. Em [91], os mesmos autores mostram que este algoritmo generalizado para problemas quadráticos convexos mantém a complexidade no pior caso.

Gay [32] mostra uma forma de obter uma sequência de soluções primais e duais factíveis em uma variante do método proposto por Karmarkar [53] para a forma padrão.

2.4 Método Primal Afim Canalizado

Uma das grandes vantagens do método simplex é a possibilidade do tratamento das variáveis canalizadas implicitamente, sem aumentar a dimensão da matriz de restrições. Neste item será desenvolvido um algoritmo primal afim que trabalha com variáveis canalizadas, sem aumentar a dimensão do problema.

O problema de PL canalizado pode ser escrito genericamente como:

$$(PC)\ \text{Min } cx$$

$$\text{sa } Ax = b$$

$$0 \leq x \leq u$$

Se uma variável x_i tiver um limite inferior $l_i \neq 0$ basta fazer uma mudança de variável $x_i' = x_i - l_i$ e o problema toma a forma (PC) apresentada acima.

O problema (PC) pode ser reescrito da seguinte forma:

$$(PC)'\ \text{Min } [c\ 0]\begin{bmatrix} x \\ w \end{bmatrix}$$

$$\begin{bmatrix} A & 0 \\ I & I \end{bmatrix}\begin{bmatrix} x \\ w \end{bmatrix} = \begin{bmatrix} b \\ u \end{bmatrix}$$

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

Vanderbei [80] apontou que, aplicando-se o algoritmo primal afim a (PC)' obtém-se uma modificação no

algoritmo que não aumenta a dimensão das matrizes envolvidas na solução do problema não canalizado; apenas altera o cálculo da matriz diagonal D.

Para o problema (PC)' a matriz AD^2A^t fica:

$$\begin{bmatrix} A & 0 \\ I & I \end{bmatrix}\begin{bmatrix} X^2 & 0 \\ 0 & W^2 \end{bmatrix}\begin{bmatrix} A^t & I \\ 0 & I \end{bmatrix} = \begin{bmatrix} AX^2 & 0 \\ X^2 & W^2 \end{bmatrix}\begin{bmatrix} A^t & I \\ 0 & I \end{bmatrix} = \left[\begin{array}{c|c} AX^2A^t & AX^2 \\ \hline X^2A^t & X^2+W^2 \end{array}\right]$$

Onde $D^2 = \begin{bmatrix} X^2 & 0 \\ 0 & W^2 \end{bmatrix}$; $X = Diag(x_1,\ldots,x_n)$ e $W = Diag(w_1,\ldots,w_n)$

As variáveis duais (y,z) são dadas por (A1.2):

$$\left[\begin{array}{c|c} AX^2A^t & AX^2 \\ \hline X^2A^t & X^2+W^2 \end{array}\right]\begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} A & 0 \\ I & I \end{bmatrix}\begin{bmatrix} X^2 & 0 \\ 0 & W^2 \end{bmatrix}\begin{bmatrix} c \\ 0 \end{bmatrix} = \begin{bmatrix} AX^2c \\ X^2c \end{bmatrix}$$

Temos portanto o seguinte sistema de equações:

$$(i)\quad AX^2A^ty + AX^2z = AX^2c$$

$$(ii)\quad X^2A^ty + (X^2+W^2)z = X^2c$$

de (ii) temos:

$$z = (X^2+W^2)^{-1}(X^2c - X^2A^ty)$$

Substituindo z em (i) temos:

$$AX^2A^ty + AX^2(X^2+W^2)^{-1}(X^2c - X^2A^ty) = X^2c$$

$$A(X^2-X^2(X^2+W^2)^{-1}X^2)A^ty = A(X^2-X^2(X^2+W^2)^{-1}X^2)c$$

a expressão $X^2-X^2(X^2+W^2)^{-1}X^2$ pode ser reescrita como:

$$X^2(X^2+W^2)^{-1}(X^2+W^2)-X^2(X^2+W^2)^{-1}X^2$$

ou seja,

$$X^2(X^2+W^2)^{-1}(X^2+W^2-X^2) = X^2(X^2+W^2)^{-1}W^2 \equiv D^*$$

Portanto

$$AD^*A^t y = AD^*c \therefore y = (AD^*A^t)^{-1}AD^*c \qquad \text{(iii)}$$

No caso do algoritmo primal canalizado, a direção é dada por:

$$\begin{bmatrix} dx \\ dw \end{bmatrix} = -\begin{bmatrix} X^2 & 0 \\ 0 & W^2 \end{bmatrix}\left(\begin{bmatrix} c \\ 0 \end{bmatrix} - \begin{bmatrix} A^t & I \\ 0 & I \end{bmatrix}\begin{bmatrix} y \\ z \end{bmatrix}\right)$$

$$\begin{bmatrix} dx \\ dw \end{bmatrix} = -\begin{bmatrix} X^2 & 0 \\ 0 & W^2 \end{bmatrix}\left(\begin{bmatrix} c \\ 0 \end{bmatrix} - \begin{bmatrix} A^t y + z \\ z \end{bmatrix}\right)$$

$$\begin{bmatrix} dx \\ dw \end{bmatrix} = -\begin{bmatrix} X^2 c - X^2 A^t y - X^2 z \\ -W^2 z \end{bmatrix}$$

Da definição do problema (PC)' $x + w = u$, logo, $dx + dw = 0 \Longrightarrow dx = -dw$. Isto pode ser verificado facilmente usando a relação (ii) e a direção dx acima. Portanto:

$$dw = W^2 z$$

$$dx = -W^2 z$$

Substituindo novamente pelo valor de z, temos:

$$dx = -W^2(X^2+W^2)^{-1}X^2(c - A^t y) = -D^{*^t}(c - A^t y)$$

Uma vez que D^* é simétrica temos:

$$dx = -D^*(c - A^t y) \qquad \text{(iv)}$$

Uma vez que D^* é uma matriz diagonal, podemos escrever:

$$D^*_{ii} = x_i^2 w_i^2 / (x_i^2 + w_i^2) = x_i^2(u_i - x_i)^2 / (x_i^2 + (u_i - x_i)^2)$$

As relações (iii) e (iv) mostram que o

cálculo da matriz diagonal D é a única alteração no algoritmo primal afim para a obtenção da direção dx e das variáveis duais y. Este resultado é muito importante pois simplifica bastante o tratamento de variáveis canalizadas. Naturalmente, deve-se também mudar o teste de bloqueio para impedir que as variáveis canalizadas ultrapassem o limite superior. O novo teste fica:

$$\alpha = \beta \, \mathrm{Min}_i(\, \mathrm{Max}(\, -x_i / dx_i , \, (u_i - x_i) / dx_i))$$

Anteriormente Vanderbei e outros [81] sugeriram a utilização de $D_i^i = \mathrm{Min}\,(x_i, u_i - x_i)$ que é mais fácil de calcular e tem o mesmo comportamento do resultado obtido quando x_i tende a um dos limites ($D_i^i \rightarrow 0$). Nossos experimentos iniciais mostram que o resultado deduzido pode levar a uma convergência mais rápida que este método embora nos testes realizados não obtivéssemos uma redução significativa no número de iterações.

A utilização de particionamento de matrizes e canalização permite a obtenção de muitos resultados interessantes. Em particular, no próximo item, veremos uma nova dedução do algoritmo dual afim.

2.5 Uma Visão Conjunta

Vanderbei [80] deriva vários algoritmos a

partir do primal afim tomando certos limites em variáveis canalizadas e aplicando o resultado em diversas formulações de PL. Neste item veremos o algoritmo geral, a partir do qual todos os outros serão derivados. Em particular, veremos a derivação do método dual afim a partir do primal afim.

Seja o PL na forma geral:

$$(PG) \qquad \text{Min } [c_c \; c_l] \begin{bmatrix} x_c \\ x_l \end{bmatrix}$$

$$\text{sa} \quad [A_c \; A_l] \begin{bmatrix} x_c \\ x_l \end{bmatrix} = b$$

$$0 \leq x_c \leq u$$

$$x_l \text{ irrestrito}$$

As variáveis x_l são livres. Para a derivação do algoritmo, Vanderbei as considera inicialmente canalizadas ($-r \leq x_l \leq r$) tomando o limite $r \to \infty$ na forma final da equação. Será utilizado aqui o limite inferior $-r$ (diferente de zero). Para considerar limites inferiores diferentes de zero basta substituir x por $x-l$ (l o limite inferior) no cálculo de D. Ou seja:

$$D^*_{ii} = (x_i - l_i)^2 (u_i - x_i)^2 \; / \; ((x_i - l_i)^2 + (u_i - x_i)^2)$$

Mais adiante, quando for calculado o limite ($r \to \infty$), toda dependência do limite inferior desaparecerá.

Serão utilizadas na demonstração, as seguintes identidades matriciais (uma dedução formal pode

ser vista no apêndice A).

$$\mathcal{A}\mathcal{B}(\mathcal{C}^{-1} + \mathcal{B}^t\mathcal{A}\mathcal{B})^{-1} = (\mathcal{A}^{-1} + \mathcal{B}\mathcal{C}\mathcal{B}^t)^{-1}\mathcal{B}\mathcal{C} \qquad (I1)$$

$$(\mathcal{C}^{-1} + \mathcal{B}^t\mathcal{A}\mathcal{B})^{-1} = \mathcal{C} - \mathcal{C}\mathcal{B}^t(\mathcal{A}^{-1} + \mathcal{B}\mathcal{C}\mathcal{B}^t)^{-1}\mathcal{B}\mathcal{C} \qquad (I2)$$

Usando a fórmula simplificada para D, temos:

$$D = \begin{bmatrix} D_c & \emptyset \\ \emptyset^c & D_l \end{bmatrix}$$

$$D_c = \text{Diag}(\ \text{Min}(\ x_c, u - x_c))$$

$$D_l = \text{Diag}(\ \text{Min}(\ r + x_l, r - x_l))$$

Lembrando que as variáveis duais são dadas por $y = (AD^2A^t)^{-1}AD^2c$ e que $A = [A_c \ A_l]$, temos:

$$y(r) = (A_cD_c^2A_c^t + A_lD_l^2A_l^t)^{-1}(A_cD_c^2c_c + A_lD_l^2c_l)$$

Seja $H \equiv (A_cD_c^2A_c^t + A_lD_l^2A_l^t)^{-1}$

Temos então dois termos na equação de y(r):

$$HA_cD_c^2c_c \text{ e } HA_lD_l^2c_l$$

Usando (I2) em H com:

$A_cD_c^2A_c^t \equiv \mathcal{C}^{-1}$; $A_l \equiv \mathcal{B}^t$; $D_l^2 \equiv \mathcal{A}$; temos:

$$(A_cD_c^2A_c^t)^{-1} - (A_cD_c^2A_c^t)^{-1}A_l(D_l^{-2} + A_l^t(A_cD_c^2A_c^t)^{-1}A_l)^{-1}A_l^t(A_cD_c^2A_c^t)^{-1}$$

Seja $B \equiv (A_cD_c^2A_c^t)^{-1}$, Substituindo em $HA_cD_c^2c_c$ temos:

$$\left[\ B - BA_l(D_l^{-2} + A_l^tBA_l)^{-1}A_l^tB\ \right]A_cD_c^2c_c \qquad (i)$$

Usando (I1) em $(HA_lD_l^2c_l)$ com:

$A_cD_c^2A_c^t \equiv \mathcal{A}^{-1}$; $A_l \equiv \mathcal{B}$; $D_l^2 \equiv \mathcal{C}$; o segundo termo

fica:

$$(A_c D_c^2 A_c^t)^{-1} A_l (D_l^{-2} + A_l^t (A_c D_c^2 A_c^t)^{-1} A_l)^{-1} c_l$$

Substituindo $B = (A_c D_c^2 A_c^t)^{-1}$, temos:

$$B A_l (D_l^{-2} + A_l^t B A_l)^{-1} c_l \qquad (ii)$$

Substituindo (i) e (ii) em y(r), temos:

$$y(r) = \left[B - B A_l (D_l^{-2} + A_l^t B A_l)^{-1} A_l^t B\right] A_c D_c^2 c_c + B A_l (D_l^{-2} + A_l^t B A_l)^{-1} c_l$$

O único termo que depende de r nesta fórmula é D_l^{-2}, que se anula para $r \to \infty$. Seja $y = \lim_{r\to\infty} y(r)$ então:

$$y = \left[B - B A_l (A_l^t B A_l)^{-1} A_l^t B\right] A_c D_c^2 c_c + B A_l (A_l^t B A_l)^{-1} c_l$$

$$y = B A_c D_c^2 c_c - B A_l (A_l^t B A_l)^{-1} (A_l^t B A_c D_c^2 c_c + c_l)$$

Note que o primeiro termo é exatamente o mesmo do algoritmo primal afim, que não tem variáveis livres.

A direção dx é dada por: $dx = \begin{bmatrix} dx_c \\ dx_l \end{bmatrix}$

$$dx_c = -D_c^2 (c_c - A_c^t y)$$

$$dx_l(r) = -D_l^2 (c_l - A_l^t y(r))$$

$$dx_l(r) = -D_l^2 \left[I - A_l^t B A_l (D_l^{-2} + A_l^t B A_l)^{-1}\right] \left[c_l - A_l^t B A_c D_c^2 c_c\right]$$

Escrevendo $I = (D_l^{-2} + A_l^t B A_l)\ (D_l^{-2} + A_l^t B A_l)^{-1}$, temos:

$$dx_l(r) = -D_l^2\left[D_l^{-2}(D_l^{-2} + A_l^t B A_l)^{-1}\right]\left[c_l - A_l^t B A_c D_c^2 c_c\right]$$

Novamente, somente D_l^{-2} depende de r.

Portanto:

$$\lim_{r\to\infty} dx_l(r) = dx_l = -(A_l^t B A_l)^{-1}\left[c_l - A_l^t B A_c D_c^2 c_c\right]$$

Resumindo, dado um problema na forma (PG), pode-se abordá-lo com o algoritmo A3 abaixo: seja x^0, um ponto inicial factível e $\beta \in (0,1)$

A3.

$k \leftarrow 0$

Repita

$D_i^i \leftarrow x_i w_i \;/\; (x_i^2 + w_i^2)^{1/2}$; $(x_i \in x_c)$ A3.1

$B \leftarrow (A_c D_c^2 A_c^t)^{-1}$ A3.2

$dx_l \leftarrow -(A_l^t B A_l)^{-1}\left[c_l - A_l^t B A_c D_c^2 c_c\right]$ A3.3

$y \leftarrow B A_c D_c^2 c_c - B A_l dx_l$ A3.4

$dx_c \leftarrow -D_c^2(c_c - A_c^t y)$ A3.5

$\alpha \leftarrow \beta \mathrm{Min}_c(\mathrm{Max}(-x_i/dx_i, (u_i - x_i)/dx_i))$ A3.6

$x^{k+1} = x^k + \alpha dx$ A3.7

$k \leftarrow k + 1$

até convergir

Consideremos novamente o problema (D) dual da forma padrão:

$$(D) \qquad \text{Max } by$$
$$\text{sa } A^t y + v = c$$
$$y \text{ irrestrito}, \; v \geq 0$$

Para aplicarmos o algoritmo A3 acima, teremos:

$A_c \equiv I$; $c_c \equiv 0$; $A_l \equiv A^t$; $c_l \equiv b$; então $B = \text{Diag}(1/v_1, \ldots, 1/v_n)$ invertendo o sentido das direções, já que (D) é um problema de maximização, temos:

$$dy = (AD^2A^t)^{-1}b$$

$$x = D^2A^t dy$$

$$dv = -D^{-2}x = -A^t dy$$

Estas são as direções (dy, dv) deduzidas para o algoritmo dual utilizando a minimização dos desvios da condição de folgas complementares. Ou seja, conforme havíamos antecipado, *o algoritmo dual afim e' na verdade, o próprio algoritmo primal afim aplicado a variáveis irrestritas.*

2.6 Método Dual Afim Canalizado

O algoritmo A3, deduzido no item anterior, abre caminho para deduções de algoritmos válidos em uma grande quantidade de problemas particulares de (PG). Neste item, veremos como obter o algoritmo dual afim com variáveis primais canalizadas sem aumentar a dimensão das matrizes envolvidas na solução do problema. No próximo item, veremos

a aplicação do algoritmo para o processo de inicialização.

Seja o PL canalizado (PC) e seu dual (DC):

(PC) $\quad \text{Min } [c_c \; c_l]\begin{bmatrix} x_l \\ x_c \end{bmatrix}$

$\text{sa } [A_l \; A_c]\begin{bmatrix} x_l \\ x_c \end{bmatrix} = b$

$0 \le x_l$

$0 \le x_c \le u$

(DC) $\quad \text{Max } by - uz$

$\text{sa } A_c^t y - z + v_c = c_c$

$A_l^t y \quad + v_l = c_l$

y irrestrito; $z, v \ge 0$

A matriz de restrições de (DC) é dada por:

$$\begin{bmatrix} A_c^t & -I & I & 0 \\ A_l^t & 0 & 0 & I \end{bmatrix}\begin{bmatrix} y \\ z \\ v_l \\ v_c \end{bmatrix} = \begin{bmatrix} c_c \\ c_l \end{bmatrix}$$

Aplicando (A3) novamente com:

$c_l \equiv b$; $c_c \equiv [-u \; 0 \; 0]$; $x_l \equiv y$; $A_l \equiv A^t$; $A_c \equiv \begin{bmatrix} -I & I & 0 \\ 0 & 0 & I \end{bmatrix}$; $x_c \equiv (z, v)$

temos $B^{-1} = \begin{bmatrix} -I & I & 0 \\ 0 & 0 & I \end{bmatrix}\begin{bmatrix} Z^2 & 0 & 0 \\ 0 & V_c^2 & 0 \\ 0 & 0 & V_l^2 \end{bmatrix}\begin{bmatrix} -I & 0 \\ I & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} -Z^2 & V_c^2 & 0 \\ 0 & 0 & V_l^2 \end{bmatrix}\begin{bmatrix} -I & 0 \\ I & 0 \\ 0 & I \end{bmatrix}$

então,

$$B^{-1} = \begin{bmatrix} Z^2 + V_c^2 & 0 \\ 0 & V_l^2 \end{bmatrix};$$

Seja $D^* \equiv B$, uma vez que B é uma matriz diagonal, D^* pode ser escrito na forma $D^* = \begin{bmatrix} D_c^* & 0 \\ 0 & D_l^* \end{bmatrix}$,

onde $D_c^* = \text{diag}(1 / (v_i^2 + z_i^2))$ e $D_l^* = \text{diag}(1/v_i^2)$

Invertendo o sentido das direções (pois o problema é de maximização) e lembrando que $dy = dx_l$, temos:

$$dy = (AD^{*}A^{t})^{-1}\left(b - AD^{*}\begin{bmatrix}-I & I & 0\\ 0 & 0 & I\end{bmatrix}\begin{bmatrix}Z^{2} & 0 & 0\\ 0 & V_{c}^{2} & 0\\ 0 & 0 & V_{l}^{2}\end{bmatrix}\begin{bmatrix}-u\\ 0\\ 0\end{bmatrix}\right)$$

$$dy = (AD^{*}A^{t})^{-1}\left(b - AD^{*}\begin{bmatrix}-Z^{2} & V_{c}^{2} & 0\\ 0 & 0 & V_{l}^{2}\end{bmatrix}\begin{bmatrix}-u\\ 0\\ 0\end{bmatrix}\right)$$

$$dy = (AD^{*}A^{t})^{-1}(b - A_{c}D_{c}^{*}Z^{2}u)$$

Para encontrarmos as variáveis duais de (DC), substituimos x por y em A3.4. Assim:

$$x = D^{*}\begin{bmatrix}-I & I & 0\\ 0 & 0 & I\end{bmatrix}\begin{bmatrix}Z^{2} & 0 & 0\\ 0 & V_{c}^{2} & 0\\ 0 & 0 & V_{l}^{2}\end{bmatrix}\begin{bmatrix}-u\\ 0\\ 0\end{bmatrix} + D^{*}A^{t}dy$$

$$\begin{bmatrix}x_{c}\\ x_{l}\end{bmatrix} = \begin{bmatrix}D_{c}^{*} & 0\\ 0 & D_{l}^{*}\end{bmatrix}\begin{bmatrix}-Z^{2} & V_{c}^{2} & 0\\ 0 & 0 & V_{l}^{2}\end{bmatrix}\begin{bmatrix}-u\\ 0\\ 0\end{bmatrix} + \begin{bmatrix}D_{c}^{*} & 0\\ 0 & D_{l}^{*}\end{bmatrix}\begin{bmatrix}A_{c}^{t}\\ A_{l}^{t}\end{bmatrix}dy$$

$$x_{c} = D_{c}^{*}(Z^{2}u + A_{c}^{t}dy)$$

$$x_{l} = D_{l}^{*}A_{l}^{t}dy$$

Para o cálculo da direção das variáveis restritas, utiliza-se A3.5 com

$$dx = \begin{bmatrix}dz\\ dv\end{bmatrix} = \begin{bmatrix}dz\\ dv_{c}\\ dc_{l}\end{bmatrix} \text{ logo,}$$

$$\begin{bmatrix}dz\\ dv_{c}\\ dc_{l}\end{bmatrix} = \begin{bmatrix}Z^{2} & 0 & 0\\ 0 & V_{c}^{2} & 0\\ 0 & 0 & V_{l}^{2}\end{bmatrix}\left(\begin{bmatrix}-u\\ 0\\ 0\end{bmatrix} - \begin{bmatrix}-I & 0\\ I & 0\\ 0 & I\end{bmatrix}\begin{bmatrix}x_{c}\\ x_{l}\end{bmatrix}\right)$$

$$dz = -Z^{2}(u - x_{c})$$

$$dv_{c} = -V_{c}^{2}x_{c}$$

$$dv_{l} = -V_{l}^{2}x_{l}$$

Resumindo, dado um problema na forma (DC), (y^0, z^0) um ponto inicial factível e $\beta \in (0,1)$, faça:

A4.

$k \leftarrow 0$

Repita

$v_c \leftarrow c_c - A_c^t y + z$

$v_l \leftarrow c_l - A_l^t y$ A4.1

$D_c^* \leftarrow \text{diag}(1 / (v_i^2 + z_i^2))$

$D_l^* = \text{diag}(1/v_i^2)$ A4.2

$dy \leftarrow (AD^*A^t)^{-1}(b - A_c D_c^* Z^2 u)$ A4.3

$x_c \leftarrow D_c^*(Z^2 u + A_c^t dy)$

$x_l \leftarrow D_l^* A_l^t dy$ A4.4

$dz = Z^2(x_c - u)$ A4.5

$dv = -V^2 x$ A4.6

$\alpha \leftarrow \beta \, \text{Min}_i(\text{Min}(-v_i/dv_i, -z_i/dz_i))$ A4.7

$y^{k+1} = y^k + \alpha dy$ A4.8

$z^{k+1} = z^k + \alpha dz$ A4.9

$k \leftarrow k + 1$

até convergir

Assim como acontece no primal afim canalizado, aqui também não há aumento na dimensão do problema. O algoritmo A4. e o primal afim canalizado foram implementados e testados numa grande quantidade de problemas. Detalhes sobre as implementações serão vistos no capítulo 3 e os resultados apresentados no capítulo 4.

2.7 Inicialização

2.7.1 Chute Inicial

Até aqui, todos os algoritmos apresentados assumem como dado de entrada um ponto inicial factível. Neste item discutiremos os métodos mais eficientes para encontrar uma solução inicial.

Vale lembrar que encontrar um ponto estritamente interior é mais fácil, em geral, que encontrar uma solução básica factível necessária para aplicação do simplex. Vários autores apresentam métodos de inicialização para o algoritmo.

A idéia por trás de todos os métodos de inicialização vistos aqui é, num certo sentido, a mesma dos métodos que procuram uma solução básica factível no algoritmo simplex, ou seja, modifica-se o problema de tal forma que se possa aplicar o próprio método ao problema modificado, para encontrar um ponto interior factível do problema original. Assim, analogamente ao simplex, denominaremos o processo de inicialização de fase I. Anstreicher [87] apresenta um algoritmo que procura uma solução factível ao mesmo tempo que busca a solução ótima do problema. Este método acrescenta uma coluna densa na matriz.

O problema modificado necessita também de um ponto interior factível. A sugestão dada por Adler e outros [01] para o método dual afim tem obtido bons resultados. Ela consiste em chutar uma solução inicial (para a fase I) em função dos dados do problema, da seguinte forma:

$$y^o = b \, \|c\| \, / \, \|A^t b\|$$

Se y^o não for interior ao politopo dual ($v^o \leq 0$), é proposto um método semelhante ao M grande ("big M") do algoritmo simplex, que será descrito mais adiante. Note que $\|A^t y^o\| = \|c\|$.

Estendemos esta idéia a outras formas de PL. Ressalta-se que o número total de iterações das fases I e II é sensível ao ponto inicial calculado. Nas heurísticas de inicialização abaixo, procura-se obter pontos que forneçam um bom valor inicial da função objetivo. Neste sentido, adotamos as inicializações apresentadas a seguir:

Para o problema primal (P): SE $c_i < 0$ $\quad x_i^c \leftarrow -c_i \, \|b\| \, / \, \|Ac\|$

SE $c_i \geq 0$ $\quad x_i^c \leftarrow \|b\| \, / \, \|Ac\|$

Primal canalizado (PC): SE $c_i > 0$ $\quad x_i^c \leftarrow u_i - c_i \, \|b\| \, / \, \|Ac\|$

SE $c_i \leq 0$ $\quad x_i^c \leftarrow u_i - \|b\| \, / \, \|Ac\|$

No caso de $x_i^c \leq 0$, fazemos: $x_i^c \leftarrow \|b\| \, / \, \|Ac\|$;

No caso de $x_i^c \geq u_i$, fazemos: $x_i^c \leftarrow \beta u_i$; $\quad \beta \cong 1$

Dual canalizado (DC): $\quad y^o = b \, \|c\| \, / \, \|A^t b\|$

$$z_i^c = 1 \, / \, u_i$$

Um resultado importante, que pode ser deduzido do algoritmo dual afim canalizado, ocorre quando todas as variáveis primais são canalizadas. Neste caso, *sempre podemos encontrar um ponto dual factível sem necessidade de fase I*. Para tanto considere as restrições matriciais do problema dual canalizado:

$$A^t y - z + v = c$$

atribui-se um valor para y^o, por exemplo, através da heurística proposta por Adler e outros descrita acima.

Quando $A_i^t y^o \geq c_i$ faz-se $z_i^c \leftarrow A_i^t y^o - c_i + 1 / u_i$ garantindo que $v^o > 0$. Caso contrário, quando $A_i^t y^o < c_i$, $z_i^c \leftarrow 1 / u_i$, como sugerido acima, ou qualquer outro número positivo.

Outra forma de inicialização, que evita a fase I no dual (com variáveis canalizadas ou não), ocorre quando $c > 0$. Neste caso, fazemos $y^o = 0$, $z^o > 0$ (qualquer valor maior que zero), $v_c^o = c_c + z^o$, e $v_l^c = c_l$.

Exceto nos casos acima, ou em certas aplicações específicas de PL, não é garantido que o ponto inicial calculado seja estritamente interior ao politopo de restrições, sendo necessária a fase I.

Estudaremos a seguir duas formas de fase I -

as baseadas em métodos de M grande, e as baseadas em variáveis livres.

2.7.2 Método M Grande Para o Algoritmo Primal Afim

Este método foi proposto por vários autores com pequenas alterações [17,34,81]. Ele consiste em acrescentar uma variável artificial ($x_{n+1} \geq 0$) com um custo "infinito" ($c_{n+1} = M$) e a respectiva coluna na matriz de restrições $A^{n+1} = b - Ax^{o}$. Fazendo-se $x^{o}_{n+1} = 1$, temos:

$$Ax^{o} + A^{n+1}x^{o}_{n+1} = b.$$

Ao aplicarmos o algoritmo primal afim a este problema, a variável artificial deve se anular no ótimo, caso contrário o problema original será infactível. Quando a variável artificial for igual a zero, teremos uma solução factível inicial para o problema original.

Este método causa um sério problema em implementações que levam em conta a esparsidade das matrizes A e $AD^{2}A^{t}$, pois, ao acrescentarmos uma coluna densa na matriz A, a matriz produto resultante $AD^{2}A^{t}$ não terá nenhum elemento estruturalmente nulo (vide capítulo 3), qualquer que seja a distribuição dos elementos não nulos da matriz A. No capítulo 3, veremos que a consideração da esparsidade das matrizes é fundamental na implementação de um algoritmo

eficiente. Portanto, este método de inicialização deve ser evitado em implementações que levam em conta a esparsidade.

Mostraremos mais adiante que será possível evitar este problema através de considerações teóricas adicionais.

2.7.3 Método M Grande Para o Algoritmo Dual Afim

Este método, proposto por Adler e outros [01], é bastante semelhante ao método M grande para o primal afim. Ele consiste em acrescentar uma variável artificial dual (y_{m+1}) irrestrita com custo "infinito" ($b_{m+1} = -M$), e a respectiva coluna na matriz transposta ($(A^t)^{m+1} = -e$), onde e é o vetor com todos elementos iguais a 1.

Em [01] é sugerido $y^0_{m+1} = -2 \text{ Min } (v^c_i)$;

$M = \mu |by^0| \ / \ y^0_{m+1}$; μ uma constante grande.

Aplica-se então o algoritmo dual afim até que $y_{m+1} < 0$ (ou seja, $v > 0$) e o algoritmo passa para o que chamamos de fase II, i.e., para a busca da solução ótima. Se o problema convergir com $y_{m+1} > 0$, o problema dual não tem solução factível.

Note que neste caso estamos introduzindo uma

linha cheia na matriz e não uma coluna como no problema anterior, portanto, a estrutura da matriz AD^2A^t é alterada "apenas" no acréscimo de uma linha e uma coluna densas, o que não é tão prejudicial, em problemas esparsos, como a perda completa da esparsidade que ocorre com o método M grande para o algoritmo primal afim.

Uma desvantagem deste método é que a linha acrescentada na matriz A pode ser linearmente dependente das linhas originais. Por exemplo, em um problema de transporte o método não pode ser usado na forma descrita, pois o vetor -e é linearmente dependente das outras linhas. No entanto, não é difícil contornar este problema.

Sugerimos uma alteração para o cálculo do tamanho do passo. Quando verificarmos que y_{m+1} ficará negativo, testa-se a direção $dy_{(1,\ldots,m)}$, antes de efetuarmos o passo. Se for de crescimento para a função objetivo original, este não é alterado. Caso contrário, faz-se o passo tão pequeno quanto possível para que $y_{m+1} < 0$. Assim, a fase II será iniciada com um valor de função objetivo melhor. Em nossas experiências, observamos problemas em que esta abordagem diminui o número de iterações da fase II por mais da metade.

Uma observação importante é que a única

alteração necessária nos algoritmos A1, A2 e A4 para a aplicação dos métodos de M grande, é o critério de convergência.

Existe uma forma de calcular a direção das variáveis duais sem aumentar a dimensão do sistema a ser resolvido, considerando a linha da variável artificial implicitamente.

Considere para tanto o seguinte vetor custo para a fase I: $b = [0\ -1]$ e a partição da matriz $\begin{bmatrix} A_t \\ -e^t \end{bmatrix}$. O sistema a ser resolvido é o seguinte:

$$\begin{bmatrix} A_t \\ -e^t \end{bmatrix} D^* [A^t\ -e] \begin{bmatrix} dy \\ dy_{m+1} \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} AD^*A^t & -AD^*e \\ -e^tD^*A^t & e^tD^*e \end{bmatrix} \begin{bmatrix} dy \\ dy_{m+1} \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$AD^*A^t dy - AD^*e dy_{m+1} = 0$$

$$-e^tD^*A^t dy + e^tD^*e dy_{m+1} = -1$$

Do primeiro sistema de equações obtemos:

$$AD^*A^t dy = AD^*e dy_{m+1}$$

A componente dy_{m+1} deve ser negativa, pois a função objetivo $[0\ -1]$ deve ser maximizada. Podemos portanto, adotar qualquer valor negativo para dy_{m+1}, por exemplo -1. Assim a direção das outras variáveis será:

$$AD^*A^t dy = -AD^*e$$

Veja que ao calcularmos dy desta forma, não há necessidade de acrescentar uma linha densa na matriz A e, devido ao bloqueio, o passo efetuado é o mesmo calculado anteriormente com $M \to -\infty$.

Uma desvantagem de usar $M \to -\infty$ é que em problemas que não têm pontos interiores, este método de inicialização não leva ao ponto ótimo. Isto pode ser contornado, fazendo-se uma "fase II", acrescentando a variável artificial, assim que for detectado heuristicamente que o problema não tem ponto interior.

2.7.4 Método da Variável Livre Para o Primal Afim

Este método desenvolvido por Vanderbei [80] evita o problema da perda da estrutura esparsa da matriz AD^2A^t, que ocorre no método de M grande para o primal afim. Ele é baseado no algoritmo geral A3 desenvolvido no item 2.5.

Considere o seguinte problema:

$$\begin{aligned} (P1)\ \text{Min}\ & \chi \\ \text{sa}\ & [A\ \rho]\begin{bmatrix} x \\ \chi \end{bmatrix} = b \\ & 0 \le x \le u \\ & \chi \text{ livre} \end{aligned}$$

$$\text{Onde } \rho = b - Ax^k;\ 0 \le x^k \le u$$

Observe que x^k é factível de (P1) com $\chi = 1$. Se encontrarmos um par (x^k, χ) tal que $\chi = 0$, então x^k é factível de (PC). Se encontrarmos o ótimo de (P1) um $\chi^* > 0$, então (PC) é infactível.

O método consiste em aplicar o algoritmo geral A3 a este problema até que o passo encontrado leve a um $\chi < 0$. Quando isto ocorrer, basta utilizar um passo (α) tal que $\chi = 0$, determinando um ponto factível inicial para (PC). Se o algoritmo convergir com $\chi > 0$, o problema (PC) é infactível.

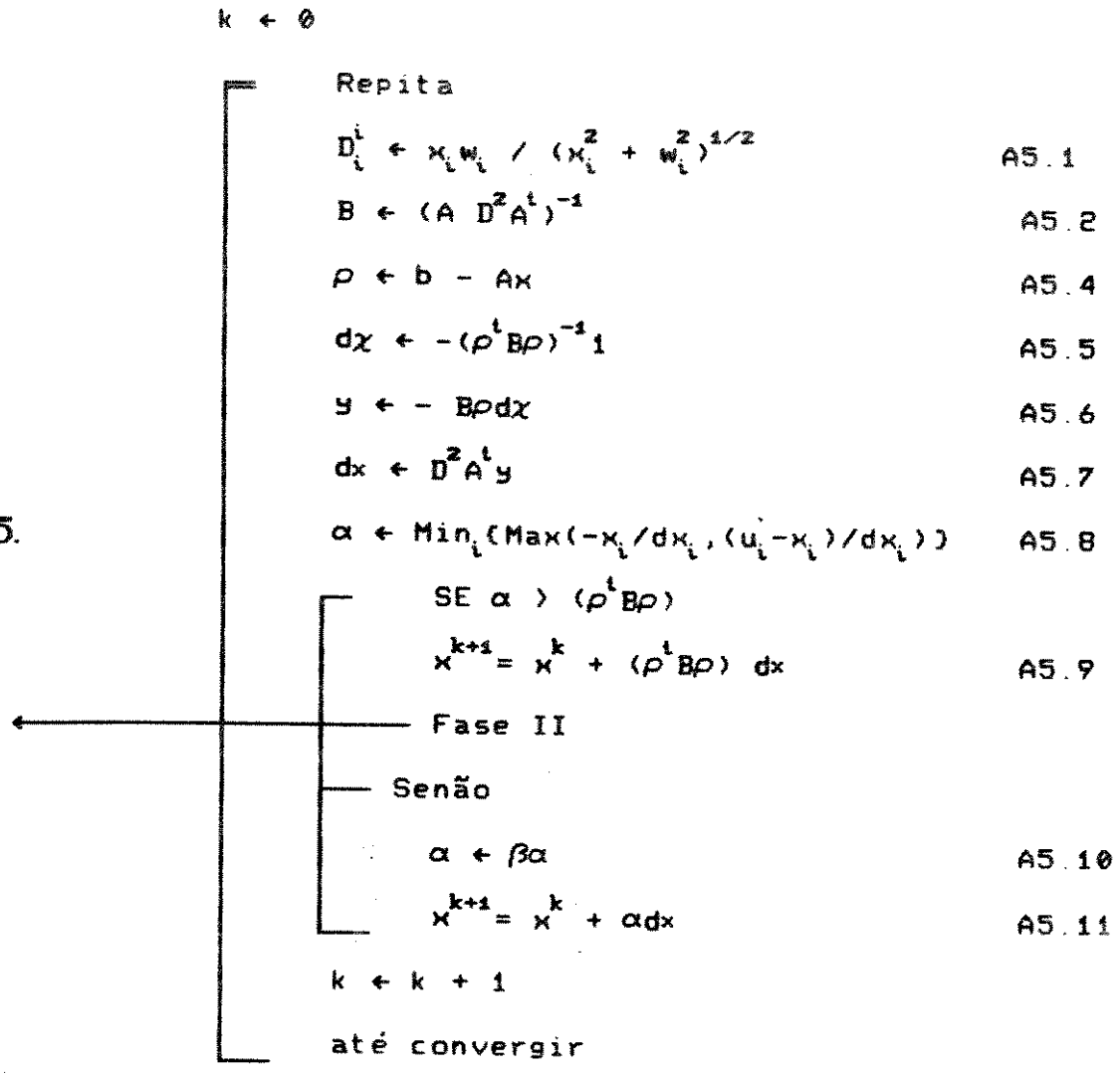
Veja que o problema difere do método M grande, quando $M \to \infty$, apenas no fato de χ ser irrestrito.

Aplicando A3 em (P1), temos:

$$c_c \equiv 0;\ c_l \equiv 1;\ A_c \equiv A;\ A_l \equiv \rho;\ B \equiv (AD^2A^t)^{-1};\ x_l \equiv \chi;\ x_c \equiv x;$$

Logo, dado um problema na forma (P1), x^0 um ponto inicial factível e $\beta \in (0,1)$ faça:

A5.



Note que $(\rho^t B\rho)^{-1}$ é um escalar, portanto não envolve a resolução de um sistema dado que B já esteja calculado. Quando $\alpha = (\rho^t B\rho)$ o produto $\alpha d\chi$ é igual a -1 e a variável artificial χ vai a zero determinando uma solução factível para (PC). Observe que a cada iteração, o problema se altera ($\rho = b - Ax^k$) e χ sempre vale 1. Vanderbei [80] sugere fazer $d\chi = -1$ e calcular os outros valores de acordo,

esta sugestão simplifica as operações.

A vantagem deste método é que, ao ser χ irrestrito, a estrutura da matriz AD^2A^t não é alterada, preservando-se todas as propriedades do problema original.

Mostraremos agora que o método de M grande visto no item 2.7.2, gera a mesma sequência de passos (quando $M \to \infty$) que o método visto neste item.

Neste caso temos $c = [0\ 1]$ e a matriz $[A\ \rho^t]$; as variáveis duais são dadas por (note que $\chi = 1$):

$$[A\ \rho^t]\begin{bmatrix} D^* & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} A^t \\ \rho \end{bmatrix} y = [A\ \rho^t]\begin{bmatrix} D^* & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$[AD^*\ \rho^t]\begin{bmatrix} A^t \\ \rho \end{bmatrix} y = \rho^t$$

$$(AD^*A^t + \rho^t\rho)y = \rho^t$$

Aplicando a atualização de "rank" 1 (apêndice A) na matriz da equação acima, temos:

$$(AD^*A^t + \rho^t\rho)^{-1} = (AD^*A^t)^{-1} - (AD^*A^t)^{-1}\rho^t(I + \rho(AD^*A^t)^{-1}\rho^t)^{-1}\rho(AD^*A^t)^{-1}$$

Aplicando (I1) com $I \equiv \mathscr{C}^{-1}$; $\rho \equiv \mathscr{B}^t$; $(AD^*A^t)^{-1} \equiv \mathscr{A}$; temos:

$$(AD^*A^t + \rho^t\rho)^{-1} = (AD^*A^t)^{-1} - (AD^*A^t + \rho^t\rho)^{-1}\rho^t\rho(AD^*A^t)^{-1}$$

pós-multiplicando por ρ^t vem:

$$(AD^*A^t + \rho^t\rho)^{-1}\rho^t = (AD^*A^t)^{-1}\rho^t - (AD^*A^t + \rho^t\rho)^{-1}\rho^t\rho(AD^*A^t)^{-1}\rho^t$$

ou seja,

$$y = y' - y\rho y'$$

onde $y' = (AD^{*}A^{t})^{-1}\rho^{t}$

finalmente isolando y:

$$y(1 + \rho y') = y'$$

$$y = y' \; / \; (1 + \rho y')$$

A direção dx é dada por:

$$\begin{bmatrix} dx \\ d\chi \end{bmatrix} = -\begin{bmatrix} D^{*} & 0 \\ 0 & 1 \end{bmatrix}\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} A^{t} \\ \rho \end{bmatrix} y\right)$$

$$dx = D^{*}A^{t}y$$

Como pode-se ver, as variáveis duais podem ser obtidas sem acrescentar explicitamente uma coluna na matriz A. Mais ainda y e y' calculados acima são vetores proporcionais às variáveis duais y calculadas no passo A5.6, e, consequentemente, as direções geradas são proporcionais também, o que determina uma mesma sequência de passos para ambos os métodos.

2.7.5 Método da Variável Livre Para o Dual Afim

Como na situação anterior, este método é baseado na introdução de uma variável livre χ e na aplicação do algoritmo A3 [80]. Considere o seguinte problema dual canalizado:

$$\text{Min } \chi$$

$$\begin{aligned} \text{sa} \quad & A_l^t y - z + v_l + \rho_l \chi = c_l \\ & A_c^t y \qquad + v_c + \rho_c \chi = c_c \\ & y, \chi \text{ livres; } z, v \geq 0 \end{aligned}$$

Onde $\rho = \begin{bmatrix} \rho_l \\ \rho_c \end{bmatrix} = \begin{bmatrix} c_l - A_l^t y + z - v_l \\ c_c - A_c^t y \qquad - v_c \end{bmatrix}$

Aplicar o algoritmo A3 a este problema equivale a aplicar o algoritmo A4 com:

$$b = [0 \; 1]; \; u = 0; \; A^t = [A^t \; \rho];$$

logo, lembrando que é necessário inverter o sinal das direções pois o algoritmo A4 é de maximização, temos (passo A4.3):

$$\begin{bmatrix} AD^* A^t & AD^* \rho \\ \rho^t D^* A^t & \rho^t D^* \rho \end{bmatrix} \begin{bmatrix} dy \\ d\chi \end{bmatrix} = - \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

uma vez que, pelas dimensões do problema, $d\chi$ é um escalar, podemos escolher seu valor como -1. Assim o primeiro conjunto de equações fica:

$$(AD^* A^t) dy - AD^* \rho = 0$$

$$dy = (AD^* A^t)^{-1} AD^* \rho$$

Obviamente esta é uma direção de factibilização pois $d\chi = -1$
As outras direções ficam:

$$dz = Z^2 D_c^* (A_c^t dy - \rho_c)$$

$$dv = -V^2 D^* (A^t dy - \rho)$$

Resumindo seja (y^0, z^0, v^0), um ponto inicial factível e $\beta \in (0,1)$

A6.

$k \leftarrow 0$

Repita

$\rho_c \leftarrow c_c - A_c^t y + z - v_c$

$\rho_l \leftarrow c_l - A_l^t y - v_l$ — A6.1

$D_c^* \leftarrow \text{diag}(1 \div (v_i^2 + z_i^2))$

$D_l^* = \text{diag}(1/v_i^2)$ — A6.2

$dy \leftarrow (AD^*A^t)^{-1} AD^* \rho$ — A6.3

$dz = Z^2 D_c^* (A_c^t dy - \rho_c)$ — A6.4

$dv = -V^2 D^* (A^t dy - \rho)$ — A6.5

$\alpha \leftarrow \text{Min}_i(\text{Min}(-v_i/dv_i, -z_i/dz_i))$ — A6.6

SE $\alpha > 1$

$y^{k+1} = y^k + dy$ — A6.7

$v^{k+1} = v^k + dv$ — A6.8

$z^{k+1} = z^k + dz$ — A6.9

Fase II

Senão

$\alpha \leftarrow \beta\alpha$ — A6.10

$y^{k+1} = y^k + \alpha dy$ — A6.11

$v^{k+1} = v^k + \alpha dv$ — A6.12

$z^{k+1} = z^k + \alpha dz$ — A6.13

$k \leftarrow k + 1$

até convergir

Problema dual infactível.

Novamente este algoritmo tem a vantagem de não modificar a estrutura das matrizes A e $AD^{*}A^{t}$, mas, por outro lado, tem duas desvantagens: primeiro, leva em geral mais iterações para encontrar um ponto factível que o algoritmo M grande para o dual afim, além disto observamos problemas numéricos no cálculo das variáveis de folga para o início da fase II (quando $\chi = 0$).

2.7.6 Diminuição do Erro de Arredondamento no Primal Afim

Já foi levantado o problema de preservar a condição $Ax^{k} = b$ no algoritmo primal afim. Chandru & Kochar [20] sugerem um método para diminuir o erro de arredondamento a cada iteração.

Considere novamente $\rho = b - Ax$ e seja d_x a direção *definida* como $dx = -D^{2}A^{t}(AD^{2}A^{t})^{-1}\rho$, então:

$$\rho^{k+1} = b - A(x^{k} + \alpha_m D^{2}A^{t}(AD^{2}A^{t})^{-1}\rho^{k})$$

$$\rho^{k+1} = b - Ax^{k} - \alpha_m AD^{2}A^{t}(AD^{2}A^{t})^{-1}\rho^{k}$$

$$\rho^{k+1} = b - Ax^{k} - \alpha_m \rho^{k}$$

$$\rho^{k+1} = \rho^{k} - \alpha_m \rho^{k}$$

Note que se $Ax^{k} = b \Longrightarrow \rho^{k} = \rho^{k+1} = 0$

$$\rho^{k+1} = (1 - \alpha_m)\rho^{k}$$

Vamos procurar α_m que minimiza $(\rho^{k+1})^{t}\rho^{k+1}$ tal que $0 \leq x^{k} \leq u$

$(\rho^{k+1})^{t}\rho^{k+1} = (1 - \alpha_m)^{2}(\rho^{k})^{t}\rho^{k}$, então:

$$\frac{d\,(\rho^{k+1})^{t}\rho^{k+1}}{d\alpha_m} = 2(1 - \alpha_m)(\rho^{k})^{t}\rho^{k} = 0 \Longrightarrow \alpha_m = 1;$$

finalmente α = Min (1, β Max($-x_i/dx_i$, $(u_i-x_i)/dx_i$))

Observe que quando α = 1 o erro vai a 0 (ρ^{k+1} = 0).

Veja que o cálculo de dx pode ser obtido aproveitando a resolução do sistema da própria iteração pois AD^2A^t já está calculado. Como a resolução é o passo que absorve o maior esforço computacional em cada iteração, o fato de se usar a mesma matriz para o cálculo de dx é extremamente conveniente. Isto será visto em detalhe no capítulo 3.

Outro fato interessante é que a direção definida é a mesma obtida no algoritmo de inicialização para o primal afim com uma variável livre, A5. Neste contexto é como se aplicássemos uma iteração da "fase I" a cada iteração da fase II. Aplicar mais de uma iteração de fase I na fase II implica em calcular novamente AD^2A^t. Note que apenas com um passo de tamanho 1 (ρ^{k+1} = 0) a variável livre se anula.

Embora o esforço computacional para o cálculo da direção seja pequeno, a utilização desta idéia em toda iteração é desnecessária. Em uma grande quantidade de problemas, o erro de Ax^k = b é insignificante, mesmo ao final da última iteração. A nossa sugestão é aplicar o algoritmo de diminuição do erro apenas quando $\|\rho\| > \varepsilon$. Em

muitos problemas esta condição não será necessária, porém, em outros, ela mostrou-se fundamental para a convergência do algoritmo primal afim. Note que esta direção pode ser de piora da função objetivo.

2.7.7 Redução da Infactibilidade

Chandru & Kochar [20] também sugerem um método de redução da infactibilidade $\rho = b - Ax^0$. Ele deve ser aplicado antes da fase I do algoritmo primal afim. O método proposto não necessita resolver um sistema linear a cada passo, que é o ponto crítico dos algoritmos já vistos.

A idéia do método é a seguinte:

Seja $\rho^k = b - Ax^k$ e seja $dx = D^2A^t\rho^k$

$$\rho^{k+1} = b - A(x^k + \alpha_m D^2 A^t \rho^k)$$

$$\rho^{k+1} = b - Ax^k - \alpha_m AD^2 A^t \rho^k$$

$$\rho^{k+1} = \rho^k - \alpha_m AD^2 A^t \rho^k$$

Vamos procurar α que minimiza $(\rho^{k+1})^t \rho^{k+1}$ tal que $0 \leq x^k \leq u$

$$(\rho^{k+1})^t \rho^{k+1} = (\rho^k - \alpha_m AD^2 A^t \rho^k)^t (\rho^k - \alpha_m AD^2 A^t \rho^k) \text{ então:}$$

$$(\rho^{k+1})^t \rho^{k+1} = (\rho^k)^t \rho^k - 2\alpha_m (\rho^k)^t AD^2 A^t \rho^k + \alpha_m^2 (\rho^k)^t (AD^2 A^t)^2 \rho^k$$

$$\frac{d((\rho^{k+1})^t \rho^{k+1})}{d\alpha} = -2(\rho^k)^t AD^2 A^t \rho^k + 2\alpha_m (\rho^k)^t (AD^2 A^t)^2 \rho^k = 0$$

então:

$$\alpha_m = \frac{(\rho^k)^t AD^2 A^t \rho^k}{(\rho^k)^t (AD^2 A^t)^2 \rho^k} = \frac{(\rho^k)^t Adx}{dx^t A^t Adx} = \frac{dx^t D^{-2} dx}{dx^t A^t Adx}$$

finalmente $\alpha = \text{Min}\ (\ \alpha_m,\ \beta \text{Max}(-x_i/dx_i, (u_i - x_i)/dx_i)\ )$

Observe que $\frac{d^2(\rho^{k+1})^t\rho^{k+1}}{d\alpha_m^2} = 2dx^t dx > 0$, portanto a direção dx é uma direção descendente.

Este algoritmo não garante a obtenção de uma solução factível para (PC), mas consegue uma redução do número de iterações da fase I. Chandru & Kochar [20] afirmam que o algoritmo pode ser usado em até O(m) iterações. Nós adotamos como critério $(\|\rho^k\| - \|\rho^{k-1}\|) / \|\rho^{k-1}\| < \varepsilon$.

2.7.8 Passo Inicial

Murty [63] sugere efetuar um passo (cuja direção é muito fácil de calcular) antes da primeira iteração para o problema no formato dual. A direção consiste apenas no gradiente da função objetivo:

$$y \leftarrow y^0 + \alpha b$$

portanto,

$$dv \leftarrow -A^t dy = -A^t b$$

logo, temos

$$\alpha = \beta \text{ Min } \{ -v_i^c / dv_i ; \text{ i.q. } dv_i < 0 \}$$

É possível verificar que a função objetivo cresce de $\alpha\|b\|^2$.

Em nossos experimentos verificamos que este procedimento não melhora significativamente o valor da função objetivo, e em alguns casos piora a convergência do

algoritmo, provavelmente, por aproximar o ponto inicial de uma restrição.

2.8 Critérios de Convergência

Sempre vale a pena lembrar que os métodos de ponto interior não levam a um ponto na fronteira do politopo de restrições. Uma vez que a solução ótima de um PL é um ponto de fronteira, é necessário definir um critério de convergência que determine uma solução com valor próximo do ótimo e que detecte esta proximidade sem a necessidade de um número muito grande de iterações nesta região. Veremos a seguir vários critérios propostos.

Adler e outros [01] propõem um critério baseado na variação da função objetivo (note que eles trabalham com o dual afim):

$$|by^k - by^{k-1}| \;/\; \text{Max}\,(\,1,\,|by^{k-1}|\,) < \varepsilon \qquad (1D)$$

Baseado no desvio da condição das folgas complementares, eles sugerem outro critério de convergência:

$$x_j^k / \|x^k\| \geq -\varepsilon_1; \quad |x_j^k v_j^k| / \|x^k\| \|v^k\| \leq \varepsilon_2 \qquad (2D)$$

Nos problemas com variáveis canalizadas, deve-se considerar também a variável de folga da canalização, $w_j = u_j - x_j$.

A extensão para o problema primal é trivial:

$$|cx^k - cx^{k-1}| \;/\; \text{Max}\,(1,\ |cx^{k-1}|) < \varepsilon \qquad \text{(1PC)}$$

$$\hat{c}_j^k \;/\; \|\hat{c}^k\| \geq -\varepsilon_1; \qquad x_j^k \hat{c}_j^k \;/\; \|x^k\|\ \|\hat{c}^k\| \leq \varepsilon_2 \qquad \text{(2PC)}$$

veja que $\hat{c} = v$.

Novamente, quando houver variáveis canalizadas, deve-se testar também $u_j - x_j$. Para tanto é necessária uma estimativa da variável dual associada à canalização (z). Esta estimativa pode ser facilmente obtida através da relação $dv = -W^2 z$ deduzida no item 2.4. Substituindo dv por -dx, temos $z = W^{-2} dx$.

Note que o critério (2PC) é a condição ótima para o problema primal afim com $\varepsilon \to 0$, como foi visto no item 2.2 [19].

Em nossas experiências, o critério (1PC) não funcionou para alguns problemas. Houve casos em que ele convergiu para uma solução muito longe da solução ótima quando consideramos $\varepsilon = 10^{-5}$. O critério (1D) teve desempenho melhor, apresentando problemas em apenas um caso de todos os testados. Nos parece que o critério mais seguro é o baseado na condição das folgas complementares, pois utiliza as próprias condições de otimalidade do problema.

Um outro critério foi proposto por Cavalier & Schall [17]. Ele faz uso da dualidade e pode ser usado tanto para o algoritmo primal quanto para o dual. O critério aqui apresentado tem uma pequena alteração para cobrir o caso onde $|cx^k| < 1$:

$$(cx^k - by^k) / \text{Max}(1, |cx^k|) < \varepsilon.$$

Deve-se usar a melhor estimativa factível da função objetivo até a iteração k.

A extensão dos critérios que utilizam a função objetivo dual para o caso de variáveis primais canalizadas é trivial, bastando substituir by por by - uz.

Vanderbei e outros [81] desenvolveram um critério que garante: $\frac{cx - cx^*}{|cx| + 1} \leq \varepsilon$. Este critério tem a desvantagem de realizar mais operações que o critério baseado na condição de folgas complementares.

Outra abordagem quanto ao critério de convergência é a obtenção de uma solução básica ótima. Este caso será visto no próximo item.

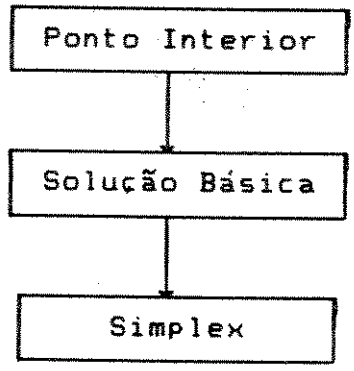
2.9 Solução Básica

Os métodos de ponto interior não encontram uma solução básica, pois trabalham com pontos estritamente interiores ao politopo. A solução ótima é obtida para uma certa precisão, como visto no item anterior. Por outro lado, existem algoritmos que, dada uma solução factível interior, encontram uma solução básica com um valor de função objetivo melhor. Esta abordagem pode ser utilizada com os seguintes propósitos:

- Encontrar uma solução básica ótima "exatamente";

- Utilizar o método simplex como um finalizador para o método de ponto interior.

O esquema abaixo ilustra estas idéias:



A alternativa de usar um "finalizador simplex" pode ser interessante, pois os algoritmos de ponto interior, tipicamente, têm uma boa convergência inicial e, a

medida que se aproximam do ponto ótimo, consegue-se variações cada vez menores na função objetivo. Resultados preliminares obtidos por Gill e outros [34] mostram que esta metodologia pode ser melhor que cada um dos métodos isoladamente (ponto interior e simplex).

Fica, no entanto, uma questão interessante em relação aos métodos mistos "ponto interior-simplex": os métodos de ponto interior surgiram dentro do contexto de encontrar um algoritmo polinomial, no pior caso, para a PL. A abordagem mista propõe uma forma de acelerar os métodos de ponto interior utilizando o algoritmo simplex que é exponencial no pior caso.

A seguir mostramos em linhas gerais um algoritmo, proposto por Benichou e outros [09], que a partir de um ponto interior, encontra uma solução básica com valor de função objetivo melhor.

A7.

Escolha as variáveis básicas e não-básicas e encontre a inversa da base;

Todas as variáveis não básicas tem "status" intermediário;

REPITA

Escolha uma variável (s) intermediária;

Atualize a coluna da variável pela inversa da base ($\hat{A}^s$);

Encontre a direção de melhora da função objetivo para esta variável ($x_s \to 0$ ou $x_s \to \infty$);

Podem ocorrer dois casos:

(i) Troca de base (encontra-se um pivô);

(ii) Variável x_s atinge um dos limites;

ATÉ não existirem variáveis intermediárias;

Observações:

- A versão do algoritmo para obter uma base dual factível pode ser facilmente obtida a partir do algoritmo A7. Neste caso seria utilizado após a obtenção da base, o algoritmo dual simplex.

- Note que cada passo do laço diminui uma variável intermediária, portanto o laço é repetido n-m vezes.

- A nossa sugestão para escolha das variáveis básicas no primeiro passo, é utilizar o conjunto de colunas linearmente independentes formado com aquelas que estão mais distantes (relativamente) de seus valores extremos (0 e u_i, quando existir limitante superiror). Em [69] é mostrado que este problema pode ser resolvido pelo método guloso.

Um fato interessante ocorre quando um problema tem solução múltipla. Neste caso os algoritmos de ponto interior não se aproximam necessariamente de uma solução básica.

Kojima [56] desenvolveu um método, baseado no algoritmo de Karmarkar, que determina se uma variável tem valor diferente de zero (0) em toda solução ótima. Este método pode determinar as variáveis básicas durante o andamento do algoritmo e a partir delas obter a solução ótima do problema. Observa-se, no entanto, que o método pode não funcionar para problemas com soluções múltiplas.

2.10 Funções não-Lineares

Este item considera brevemente algumas das diversas variações de métodos de ponto interior. As variações aqui tratadas têm em comum a utilização de funções não lineares como auxílio na obtenção das direções de

melhora da função objetivo original (linear). A seguir são mostradas algumas das funções utilizadas.

a) Função Potencial

$$\left[f(x) = n \log(cx) - \sum_{j=1}^{n} \log(x_j) \right]$$

Esta função foi introduzida por Karmarkar [53]. Para sua utilização ele assume que o valor da função objetivo no ponto ótimo vale zero ($cx^* = 0$).

Todd e Burrel [78] sugeriram a utilização das variáveis duais para uma estimativa do valor ótimo a cada passo. Assim, a função potencial seria reescrita:

$$\left[f(x) = n \log(cx-v) - \sum_{j=1}^{n} \log(x_j) \right]$$

onde v é o melhor limite inferior para cx^k obtido via uma estimativa factível das variáveis duais. Eles demonstram também que a estimativa das variáveis duais é exata no ponto ótimo. Anstreicher [04] calcula um limite inferior para o valor ótimo da função objetivo através de motivação geométrica. Mais tarde, Goldfarb e Mehrotra [36] desenvolveram uma variante polinomial do algoritmo que não necessita do valor exato de cx^*.

b) Função Barreira:

$$\left[f(x) = cx - \mu \sum_{j=1}^{n} \log(x_j) \right]$$

onde μ é conhecido como parâmetro de barreira e pode ser demonstrado que $\mu \to 0$ quando x se aproxima do ponto ótimo.

A função barreira foi proposta por Gill e outros [34], inspirada no método das barreiras [33,58] para problemas não lineares. Eles concluem que utilizando um parâmetro μ apropriado, as direções obtidas com este método são idênticas as do método projetivo [53]. Além disto, eles utilizam uma heurística para o cálculo da matriz diagonal D, quando houver variáveis canalizadas. Em [35] é desenvolvido o método das barreiras para o formato dual (D).

c) Função Potencial Logarítmica

$$\left[f(x) = q \log(cx - v) - \sum_{j=1}^{n} \log(x_j) \right]$$

Em [42], Gonzaga demonstra que esta função para $q \geq n + \sqrt{n}$ tem complexidade polinomial no pior caso. Note que para $q = n$ a função é a mesma utilizada por Karmarkar.

Diversas outras formas tem sido utilizadas como por exemplo métodos de penalidades [50,61]. A utilização de funções não lineares torna necessário um cálculo mais sofisticado no tamanho do passo. A idéia que tem ganho maior corpo é a utilização de busca unidimensional nestas funções.

Gonzaga [40] demonstra que todos os métodos surgidos geram direções que são uma combinação linear das seguintes direções:

$$c_p = P\bar{c} = PDc = -D^{-1}dx_c$$

$$e_p = Pe$$

$$\text{onde } P = I - DA^t(AD^2A^t)^{-1}AD,$$

$$e = (1,1,\ldots,1)$$

A partir desta demonstração ele sugere um algoritmo baseado em busca bidimensional, que encontra a melhor combinação destas duas direções. Os algoritmos baseados em funções lineares estudados nos itens anteriores (primal afim e dual afim) utilizam apenas a direção c_p pois $\hat{c} = D^{-1}c_p + 0e_p$.

2.11 Centragem

2.11.1 Ponto Central

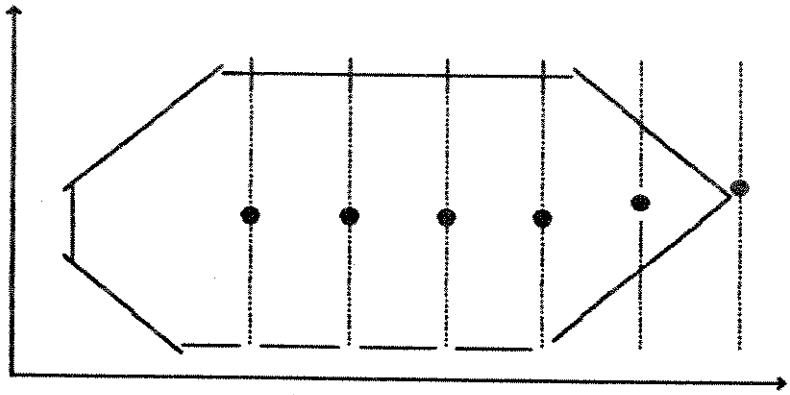
O conceito de centro visto até agora foi definido como o ponto onde o valor de todas as variáveis não negativas é o mesmo. A obtenção de pontos centrais é feita através de transformações que modificam o problema, transformando-o em um problema equivalente.

Neste item vamos usar outro conceito de centro, ou seja, o centro de um politopo é o ponto que maximiza o produto das variáveis não negativas $\left(\prod_{i=1}^{n} x_i \right)$. Por conveniência, é preferível trabalhar com a função que maximiza o somatório do logaritmo das variáveis:

$$\text{Max } f(x) = \left(\sum_{i=1}^{n} \log(x_i) \right)$$

É possível mostrar que as duas funções tem o mesmo ponto ótimo.

A partir deste conceito, definimos trajetória central como o conjunto de pontos do politopo que satisfazem esta função a custo constante. Isto é ilustrado pela figura a seguir:



Trajetória Central

Barnes e outros [07] propuseram um procedimento nos métodos de pontos interiores, que permite manter a sequência de pontos gerados pelo algoritmo próxima da trajetória central, preservando o valor da função objetivo obtida a cada iteração. A utilização deste conceito evita que os pontos se aproximem da fronteira do politopo de restrições. Desta forma espera-se que as direções geradas sejam melhor aproveitadas, podendo reduzir o número total de iterações. Em alguns problemas a utilização de centragem reduz o número de iterações a menos da metade.

Existem duas abordagens para a aplicação da centragem: utilizá-la antes ou depois da iteração do método de ponto interior, sendo a segunda delas a mais utilizada. Outra idéia consiste em realizar mais de uma centragem por iteração. Isto pode aumentar muito o tempo da iteração, mas espera-se que este aumento seja compensado por uma redução mais significativa no número total delas. Em [05] vemos uma comparação do número de iterações de vários métodos e seu comportamento em função de centragens.

Uma observação interessante é que a função barreira, vista no item anterior, é uma combinação da função objetivo com a função de centragem definida acima. Na verdade, os métodos baseados nas funções barreira e potencial, visam manter o ponto próximo da trajetória central, por este motivo, tais métodos são polinomiais.

2.11.2 Centragem Para o Método Primal Afim

Definida a função, é preciso determinar uma direção para a centragem. Uma vez que a centragem será utilizada em conjunto com o método de ponto interior primal afim, uma direção natural é obtida utilizando o próprio algoritmo primal afim para a centragem, substituindo o vetor custo c pelo gradiente da função ($\nabla f(x)$) (em [07] é utilizado o método projetivo). Desta forma teríamos:

$$dx_c = D^2(\nabla f(x) - A^t y)$$

onde dx_c é a direção de centragem;

$y = (AD^2A^t)^{-1}AD^2\nabla f(x)$;

e $\nabla f(x_i) = 1 / x_i$

O número de operações para a obtenção desta direção é o mesmo para o cálculo da direção normal do método. Por outro lado, se utilizarmos o mesmo fator de escala D_i^i, a decomposição da matriz AD^2A^t seria aproveitada para a obtenção de ambas as direções, tornando menos caro o

esforço computacional do cálculo das duas direções.

Para evitar que o valor da função objetivo piore no processo de centragem, projeta-se a direção de centragem na curva de nível da função objetivo. Isto pode ser obtido facilmente fazendo uso a direção dx, utilizada pelo método de ponto interior no último passo:

$$dx'_c = dx_c - \frac{cdx}{cdx} c \; dx$$

é fácil verificar que $cdx'_c = 0$.

2.11.1 Centragem Para o Método Primal Canalizado

A dedução a seguir é muito semelhante à utilizada no desenvolvimento do algoritmo primal afim canalizado. Considere novamente o problema:

$$\begin{aligned} (PC) \quad & \text{Max } cx \\ & \text{sa } \; Ax = b \\ & \qquad x + w = u \\ & \qquad x, w \geq 0 \end{aligned}$$

Para este problema, a função de centragem fica: $\text{Max } f(x,w) = \sum_{i=1}^{n} \log(x_i) + \sum_{i=1}^{n} \log(w_i)$

Aplicando o algoritmo primal com c substituído por $\nabla f(x,w)$, temos:

$$\begin{bmatrix} A & 0 \\ I & I \end{bmatrix}\begin{bmatrix} X^2 & 0 \\ 0 & W^2 \end{bmatrix}\begin{bmatrix} A^t & I \\ 0 & I \end{bmatrix}\begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} A & 0 \\ I & I \end{bmatrix}\begin{bmatrix} X^2 & 0 \\ 0 & W^2 \end{bmatrix}\begin{bmatrix} \nabla fx \\ \nabla fw \end{bmatrix}$$

onde $\nabla fx_i = 1 / x_i$ e $\nabla fw_i = 1 / w_i$

Realizando-se as operações, tem-se:

$$\begin{bmatrix} AX^2A^t & AX^2 \\ X^2A^t & X^2+W^2 \end{bmatrix}\begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} AX^2 & 0 \\ X^2 & W^2 \end{bmatrix}\begin{bmatrix} \nabla fx \\ \nabla fw \end{bmatrix}$$

Logo,

(i) $AX^2A^ty + AX^2z = AX^2\nabla fx$

(ii) $X^2A^ty + (X^2+W^2)z = X^2\nabla fx + W^2\nabla fw$

Isolando z em (ii), temos:

$$z = (X^2+W^2)^{-1}(X^2\nabla fx + W^2\nabla fw - X^2A^ty)$$

Substituindo z em (i) temos:

$$AX^2A^ty + AX^2(X^2+W^2)^{-1}(X^2\nabla fx + W^2\nabla fw - X^2A^ty) = AX^2\nabla fx$$

Definindo D^* como no item 2.4, ou seja:

$D_i^{i*} = Diag(x_i^2w_i^2 / (x_i^2+w_i^2))$ vem:

$$AD^*A^ty = AD^*\nabla fx - AD^*\nabla fw$$

$$y = (AD^*A^t)^{-1}AD^*(\nabla fx - \nabla fw)$$

A direção de centragem fica:

$$\begin{bmatrix} dx_c \\ dw_c \end{bmatrix} = \begin{bmatrix} X^2 & 0 \\ 0 & W^2 \end{bmatrix}\left(\begin{bmatrix} \nabla fx \\ \nabla fw \end{bmatrix} - \begin{bmatrix} A^t & I \\ 0 & I \end{bmatrix}\begin{bmatrix} y \\ z \end{bmatrix}\right)$$

$$dx_c = X^2\nabla fx - X^2A^ty - X^2z$$

$$dv_c = W^2\nabla fv - W^2 z$$

lembrando que $dx = -dv$, temos:

$$dx_c = -W^2\nabla fv + W^2 z$$

$$dx_c = -W^2\nabla fv + W^2(X^2 + W^2)^{-1}(X^2\nabla fx + W^2\nabla fv - X^2 A^t y)$$

$$dx_c = -D^*\nabla fv + D^*\nabla fx - D^* A^t y$$

$$dx_c = D^*(\nabla fx - \nabla fv - A^t y)$$

O cálculo da projeção de dx_c na curva de nível não se altera com relação ao problema não canalizado.

Portanto, a centragem para o problema canalizado pode ser feita sem o aumento das dimensões das matrizes envolvidas, assim como no algoritmo primal afim canalizado. é importante o fato da matriz diagonal D^* ser a mesma na centragem e no cálculo da direção do método primal afim, pois permite a obtenção das duas direções com um esforço menor.

2.11.4 Centragem Para o Algoritmo Dual Afim

Será considerado apenas o problema com variáveis primais canalizadas. O problema não canalizado é análogo. Para o algoritmo dual afim canalizado, a função de centragem é dada por: Max $f(v,z)$, onde:

$$f(v,z) = \sum_{i=1}^{n} \log(v_i) + \sum_{i=1}^{n} \log(z_i)$$

Note que as variáveis y não aparecem na

função de centragem, por serem irrestritas.

Para obtermos a direção de centragem, vamos aplicar o algoritmo geral A3 (item 2.5) ao problema dual canalizado (DC), substituindo o vetor custo pelo gradiente da função de centragem e trocando o sinal das direções, pois o algoritmo A3 é de minimização. Desta forma, o algoritmo tem uma dedução semelhante ao algoritmo dual afim canalizado A4. A diferença está no vetor custo: $[\nabla fz, \nabla fv]$ e $b = 0$ (pois y é irrestrito). Assim, temos novamente:

$$D^* = \begin{bmatrix} D_c^* & 0 \\ 0 & D_l^* \end{bmatrix} = \begin{bmatrix} Z^2+V_c^2 & 0 \\ 0 & V_l^2 \end{bmatrix}^{-1}$$

A direção das variáveis livres será:

$$dy_c = (AD^*A^t)^{-1}\left(0 - AD^*\begin{bmatrix} -I & I & 0 \\ 0 & 0 & I \end{bmatrix}\begin{bmatrix} Z^2 & 0 & 0 \\ 0 & V_c^2 & 0 \\ 0 & 0 & V_l^2 \end{bmatrix}\begin{bmatrix} \nabla fz \\ \nabla fv_c \\ \nabla fv_l \end{bmatrix}\right)$$

$$dy_c = -(AD^*A^t)^{-1}AD^*\begin{bmatrix} -Z^2\nabla fz + V_c^2\nabla fv_c \\ V_l^2\nabla fv_l \end{bmatrix}$$

A estimativa das variáveis primais é dada por:

$$x = \begin{bmatrix} x_c \\ x_l \end{bmatrix} = -\begin{bmatrix} D_c^* & 0 \\ 0 & D_l^* \end{bmatrix}\begin{bmatrix} -I & I & 0 \\ 0 & 0 & I \end{bmatrix}\begin{bmatrix} Z^2 & 0 & 0 \\ 0 & V_c^2 & 0 \\ 0 & 0 & V_l^2 \end{bmatrix}\begin{bmatrix} \nabla fz \\ \nabla fv_c \\ \nabla fv_l \end{bmatrix} - D^*A^t dy_c$$

ou seja,

$$x_c = -D_c^* (-Z^2 \nabla fz + V_c^2 \nabla fv_c + A^t dy_c)$$

$$x_l = -D_l^* (V_l^2 \nabla fv_l + A^t dy_c)$$

Assim as direções de centragem dz_c e dv_c ficam:

$$dz_c = Z^2 (\nabla fz - x_c)$$

$$dv_c = V^2 (\nabla fv + x)$$

É necessário projetar as direções a custo constante. Gonzaga [43] sugere (para o dual sem canalização):

$$dy'_c = dy_c - \frac{bdy}{bdy}_c dy$$

analogamente para dz_c teríamos:

$$dz'_c = dz_c - \frac{udz}{udz}_c dz$$

Pode-se verificar que $bdy'_c - udz'_c = 0$

A direção dv_c é calculada de acordo, logo:

$$dv'_{c_c} = dz'_c - A_c^t dy'_c$$

$$dv'_{l_c} = - A_l^t dy'_c$$

2.11.5 Busca Unidimensional

A função de centragem é não linear, portanto, é necessário utilizar um método de busca unidimensional para determinar o passo que obtém o melhor valor da função de centragem na direção calculada. Em [05] é sugerido o método de Newton pela sua velocidade de convergência.

Para simplificar a notação, definiremos o conjunto P de todas as variáveis não negativas ($p_i \geq 0$). Deste modo, a função de centragem para todos os casos (primal, primal canalizado, dual e dual canalizado) pode ser escrita como:

$$\text{Max } f(p) = \sum_{p_i \in P} \log(p_i)$$

Desejamos encontrar o passo α que maximiza $f(p + \alpha dp)$ no intervalo $(0, \min(-p_i / dp_i \text{ t.q. } dp_i < 0))$.

A derivada de f em relação a α é:

$$\frac{df(\alpha)}{d\alpha} = \sum_{p_i \in P} \frac{dp_i}{p_i + \alpha dp_i} = \sum_{p_i \in P} \frac{1}{p_i \div dp_i + \alpha}$$

A derivada segunda é:

$$\frac{d^2 f(\alpha)}{d\alpha^2} = - \sum_{p_i \in P} \frac{1}{(p_i \div dp_i + \alpha)^2}$$

Portanto, a equação de recursividade do método de Newton [58] será:

$$\alpha_{k+1} = \alpha_k + \frac{\sum_{p_i \in P} \frac{1}{p_i \div dp_i + \alpha_k}}{\sum_{p_i \in P} \frac{1}{(p_i \div dp_i + \alpha_k)^2}}$$

O método de Newton tem uma convergência

rápida, mas efetua muitas operações por iteração. O método da falsa posição [58] converge mais lentamente porém efetua menos operações, pois só utiliza a derivada primeira, sendo mais vantajoso na maioria dos problemas testados. A equação de recursividade deste método é mostrada a seguir:

$$\alpha_{k+1} = \alpha_k - f'(\alpha_k)\left[\frac{\alpha_k - \alpha_{k-1}}{f'(\alpha_k) - f'(\alpha_{k-1})}\right]$$

2.11.6 Centragem na Fase I

A simplicidade na dedução da direção de centragem para várias formas de PL utilizando o algoritmo A3, permite a sua aplicação em muitas situações. Em particular, aplicamos a centragem nos problemas utilizados para encontrar uma solução inicial factível para os métodos primal e dual. Os resultados obtidos são semelhantes à centragem na fase II.

Os métodos de inicialização vistos no item 2.7 consistem em resolver problemas de PL modificados em relação ao problema original. Portanto, a idéia de centragem também pode ser aplicada a estes métodos.

No algoritmo primal afim, desejamos maximizar

a mesma função de centragem do item 2.11.3, pois a variável artificial χ é irrestrita. Utilizando novamente o algoritmo geral A3 para o problema de inicialização de (PC), chega-se ao seguinte valor das variáveis duais:

$$y = (AD^{*}A^{t})^{-1}AD^{*}(\nabla fx - \nabla fv - \rho d\chi)$$

onde χ é a variável artificial e $\rho = b - Ax$

Determinando uma direção arbitrária para $d\chi$ (que é um escalar), por exemplo $d\chi = -1$, temos:

$$y = (AD^{*}A^{t})^{-1}AD^{*}(\nabla fx - \nabla fv + \rho)$$

Portanto a direção de centragem será:

$$dx_c = -D^{*}(\nabla fx - \nabla fv - A^{t}y)$$

Projetando esta direção na curva de nível da função objetivo na fase I ($c = [0\ 1]$), obtemos:

$$dx'_c = dx_c - \frac{d\chi}{d\chi_c}\, dx$$

Uma vez que $d\chi$ foi arbitrado em -1, a direção dx_c pode ser descendente em relação a função de centragem. Se for o caso, basta inverter o sinal da direção. Outra opção é fazer a busca unidimensional permitindo valores negativos para o passo α.

O cálculo da direção de centragem para o algoritmo dual é semelhante ao desenvolvido no item 2.11.4, bastando acrescentar uma linha (-e) na matriz A. Assim:

$$AD^{*}A^{t}dy + AD^{*}(-e)d\chi = - AD^{*}(V^{2}\nabla fv - Z^{2}\nabla fz)$$

$$-e^{t}D^{*}A^{t}dy + e^{t}D^{*}ed\chi = - -e^{t}D^{*}(V^{2}\nabla fv - Z^{2}\nabla fz)$$

Novamente, fazendo $d\chi = -1$ temos:

$$AD^{*}A^{t}dy = - AD^{*}(V^{2}\nabla fv - Z^{2}\nabla fz + e)$$

Assim:

$$x_{c} = -D_{c}^{*}(V_{c}^{2}\nabla fv_{c} - Z^{2}\nabla fz + A_{c}^{t}dy + e)$$

$$x_{l} = -D_{l}^{*}(V_{l}^{2}\nabla fv_{l} + A_{l}^{t}dy + e)$$

As direções ficam:

$$dz_{c} = Z^{2}(\nabla fz - x_{c})$$

$$dv_{c} = V^{2}(\nabla fv + x)$$

As considerações sobre a projeção na curva de nível feitas para a centragem na fase I primal aplicam-se também na fase I dual desenvolvida acima. Vale lembrar que a função objetivo na fase I dual é $b = [0\ -1]$.

2.12 Iterações Continuadas

Uma forma que utilizamos [23] para obter valores melhores da função objetivo em uma iteração do método dual afim, consiste em eliminar a componente da variável que bloqueia, atualizar a direção dy de acordo e, se a direção atualizada ainda for de crescimento, realizar novo passo. A operação é repetida enquanto for possível ou até um número limite de vezes [23]. Assim como a centragem, a idéia aqui é reduzir o número total de iterações.

Em exemplos de pequeno porte, esta idéia produziu resultados melhores que o algoritmo básico. Em problemas maiores, obtivemos resultados melhores ou piores, conforme o número de iterações continuadas (variáveis bloqueio eliminadas) permitido e a quantidade de iterações onde esta abordagem foi utilizada.

Outra forma que pode ser adotada seria o aproveitamento da centragem. Neste caso, realiza-se a centragem ao final da iteração e, feito isso, caminha-se novamente na mesma direção utilizada antes da centragem. O procedimento pode ser repetido algumas vezes dentro de uma mesma iteração. Esta abordagem é utilizável para ambos os métodos (primal e dual). Ela obteve os melhores resultados das variações do algoritmo básico testadas no capítulo 4.

2.13 Transformação num Problema com Restrições de Desigualdade

Gonzaga [43] propõe um método de transformar problemas na forma padrão (P), de maneira que as restrições do problema sejam escritas na forma de inequações. Considere novamente o problema na forma primal:

$$(P) \qquad \text{Min } cx$$
$$\text{sa} \quad Ax = b$$
$$x \geq 0$$

Escolhendo-se m colunas de A linearmente independentes e denominando o conjunto correspondente de variáveis básicas (x_B) e o conjunto complementar de variáveis não básicas (x_N), temos as partições correspondentes $[c_N \; c_B]$ e $[A_N \; A_B]$. O problema (P) pode ser reescrito da seguinte forma:

$$(P1) \qquad \text{Min } [c_N \; c_B] \begin{bmatrix} x_N \\ x_B \end{bmatrix}$$
$$\text{sa} \quad [A_N \; A_B] \begin{bmatrix} x_N \\ x_B \end{bmatrix} = b$$
$$x \geq 0$$

Seja $B = A_B^{-1}A_N$ e $\hat{b} = A_B^{-1}b$. Multiplicando o conjunto de restrições do problema (P1) por A_B^{-1}, temos:

$$(P2) \qquad \text{Min } [c_N \; c_B] \begin{bmatrix} x_N \\ x_B \end{bmatrix}$$

$$\text{sa} \quad [B \quad I \;] \begin{bmatrix} x_N \\ x_B \end{bmatrix} = \hat{b}$$

$$x \geq 0$$

Definindo $y = x_N$ temos $x_B = \hat{b} - By$, substituindo x_N em (P2) com:

$$cx = c_N x_N + c_B x_B = c_N y + c_B(\hat{b} - By)$$

$$cx = (c_N - c_B B)y + c_B \hat{b} = \hat{c}y + \hat{c}_o, \text{ onde}$$

$$c_N - c_B B = \hat{c}; \; c_B \hat{b} = \hat{c}_o \text{ temos:}$$

$$(P3) \qquad \text{Min } (\hat{c}y - \hat{c}_o)$$

$$\text{sa } By + x_B = \hat{b}$$

$$y \geq 0$$

$$x_B \geq 0$$

Deixando implícitas as variáveis x_B temos:

$$(P4) \qquad \text{Min } (\hat{c}y - \hat{c}_o)$$

$$\text{sa } By \leq \hat{b}$$

$$-Iy \leq 0$$

ou seja:

$$(P5) \qquad \text{Min } (\hat{c}y - \hat{c}_o)$$

$$\text{sa } \begin{bmatrix} B \\ -I \end{bmatrix} y \leq \begin{bmatrix} \hat{b} \\ 0 \end{bmatrix}$$

Esta é a mesma formulação do problema dual. A consideração da matriz identidade como parte da matriz de restrições, permite tratar as variáveis y como

"irrestritas", e as variáveis x_B fazem o papel de variáveis de folga.

Uma vantagem desta formulação é que ela torna possível relacionar as direções nos problemas (P) e (P5) [43]. Uma comparação do comportamento do método para várias funções é feita em [05].

Outro fato a destacar é que o problema (P5) trabalha com dimensões de matrizes diferentes das matrizes de (P). O sistema a ser resolvido em (P), ou no algoritmo dual que resolve (D), tem dimensão m. Em (P5) a dimensão é (n-m). Portanto esta abordagem pode ser interessante para $n-m < m$, isto é, $n < 2m$. Porém, quando $n > 2m$ não parece ser aconselhável. Outro problema que pode ocorrer é a diminuição da esparsidade do sistema ao calcularmos $B = A_B^{-1}A_N$.

2.14 Complexidade do Algoritmo

2.14.1 Hipóteses Sobre o Problema

A demonstração utilizada por Karmarkar [53] requer um formato especial para o problema e, também, faz algumas hipóteses que serão depois relaxadas. O formato do problema é o seguinte:

$$
\begin{aligned}
(P0) \quad \text{Min } & cx \\
\text{sa } & Ax = 0 \\
& e^t x = 1 \\
& x \geq 0
\end{aligned}
$$

Supõe-se ainda que

$$
\begin{aligned}
Ae &= 0; \ e \\
cx^* &= 0
\end{aligned}
$$

Além disso é necessário que a região factível seja limitada. Note que a restrição $Ae = 0$ implica que o ponto *interior* $x^o = \frac{1}{n}e$ seja factível.

Existem várias formas para se converter um problema da forma padrão (P) para a forma homogênea (P0) [53,25,36,78,79]. A forma proposta por Karmarkar [53] aumenta muito as dimensões do problema. Vimos no item 2.10 que é possível tratar problemas com valor ótimo de função objetivo desconhecido, por exemplo, através de limites inferiores obtidos via estimativa das variáveis duais. Gay [32] e Ye & Kojima [83] apresentam variantes do algoritmo que trabalham diretamente com a forma padrão (P). Rinaldi [71] apresenta uma especialização do algoritmo a partir de uma transformação do problema na forma padrão com variáveis canalizadas.

2.14.2 Tamanho do Problema

Necessitaremos, para o cálculo da

complexidade, de alguns resultados intermediários que relacionam grandezas do problema com o seu tamanho em "bits". Para tanto, formularemos outra hipótese: todos os dados de entrada (elementos de A, b e c) são números inteiros.

Seja a função comp(F) o comprimento em "bits" de uma matriz com coeficientes inteiros. Logo:

$$comp(A) = \sum_{i=1}^{m} \sum_{j=1}^{n} \log_2(|A_i^j| + 1)$$

$$comp(b) = \sum_{i=1}^{m} \log_2(|b_i| + 1)$$

$$comp(c) = \sum_{j=1}^{n} \log_2(|c^j| + 1)$$

O comprimento em bits de um PL é então:

$$L = comp(A) + comp(b) + comp(c) + comp(mn)$$

$$\text{Seja } P = \left(\prod_{j=1}^{m} Max(|b_i|, 1) \left(\prod_{j=1}^{n} Max(|A_i^j|, 1) \right) \right)$$

é fácil ver que $P > |\det [A^{B-j} \; b]| \; \forall$ conjunto B de m colunas linearmente independentes de A e $\forall \; j \in B$. Logo:

$$|\det [A^{B-j} \; b]| < P < 2^L$$

Com estas definições podemos mostrar as seguintes afirmações [41]: dado $x = [x_B \quad x_N]$, uma solução básica factível de um problema na *forma padrão* (P) com valor da função objetivo no ponto ótimo nulo e região factível

limitada, então:

$$(i)\ x_i = 0 \text{ ou } x_i > 2^{-L}\ \forall\ i \in (1,..,n)$$

$$(ii)\ cx = 0 \text{ ou } cx > 2^{-L}$$

$$(iii)\ cx \leq 2^{L},\ e^t x \leq 2^{L}\ \forall\ x \text{ factível de (P)}$$

Demostração:

$$(i)\ Ax = b$$

$$A^B x_B + A^N x_N = b \qquad (x_N = 0)$$

$$A^B x_B = b$$

$$x_{B_j} = |\det [A^{B-j}\ b] / \det(A_B)|$$

$$x_{B_j} \geq |\det [A^{B-j}\ b]| / P$$

$$x_{B_j} \geq |\det [A^{B-j}\ b]|\ 2^{-L}$$

$|\det [A^{B-j}\ b]|$ é inteiro, portanto se $|\det[A^{B-j}\ b]| = 0$ então $x_{B_j} = 0$, caso contrário $x_{B_j} \geq 2^{-L}$ □

(ii) Se $x = 0$ então $cx = 0$.

Caso contrário $cx \geq 0$ pois, $cx^* = 0$, logo:

$c_N x_N = 0$, $c_B x_B \geq 0$, mas como os elementos de c são inteiros e $x_B \geq 2^{-L}$, então: $cx \geq 2^{-L}$ ou $cx = 0$ □

(iii) seja $x = [x_B\ x_N]$ então:

$$cx \leq 2^{[comp(c) + comp(A) + comp(b)]} < 2^{L}$$

$$e^t x \leq n\|x\|_\infty \leq n2^{[comp(A) + comp(b)]}$$

$$e^t x \leq 2^{[comp(A) + comp(b) + comp(n)]} < 2^{L}$$

Uma vez que a região factível é limitada, qualquer ponto factível é uma combinação convexa dos vértices, portanto:

$$cx < 2^{L}$$
$$e^{t}x < 2^{L} \qquad \forall\ x \text{ factível}$$

□

Consequentemente, se encontrarmos uma solução com custo menor que 2^{-L}, ela é ótima (ii). Vimos, no item 2.9, um algoritmo polinomial que obtém uma solução básica com valor de função objetivo melhor, a partir de um ponto interior (algoritmo de purificação). Portanto, um método de ponto interior polinomial, associado a um algoritmo de purificação, pode encontrar uma solução básica ótima em um tempo polinomial. Por outro lado, o critério $cx < 2^{-L}$ não é prático pois 2^{-L} é um número *muito* pequeno. Na realidade adota-se tolerâncias mais flexíveis.

2.14.3 Polinomialidade do Método Projetivo

Neste item veremos que o algoritmo proposto por Karmarkar [53] para resolver (P0) é polinomial no pior caso. A seguir reproduzimos os passos do algoritmo para a demonstração: dados (P0), $\beta \in (0,1)$, $r = 1 / \sqrt{n(n-1)}$, faça:

A0.

$k \leftarrow 0$

Repita

$B \leftarrow \begin{bmatrix} AD^k \\ e^t \end{bmatrix}$ A0.1

$c_p \leftarrow (I - B^t(BB^t)^{-1}B)Dc$ A0.2

$x' \leftarrow x^k - \beta r c_p / \|c_p\|$ A0.3

$x^{k+1} \leftarrow Dx' / e^t Dx'$ A0.4

$k \leftarrow k + 1$

Até convergir

Para a demonstração, utilizaremos uma combinação de teoremas propostos por vários autores.

Lema 1 (Teorema 5 [53], Teorema 2 [65])

O ponto x' obtido no passo A0.3 é ótimo do problema

$$\text{Min } cD^k x$$

$$\text{sa } \Omega^k \cap B(x^o, \beta r)$$

onde Ω^k é o politopo formado por $AD^k x = 0$ e $e^t x = 1$; $B(x^o, \beta r)$ é a esfera com centro em e/n e raio βr.

Demonstração: A interseção $B(e/n, \beta r) \cap \Omega^k$ é uma esfera (para $\beta r \leq 1 / \sqrt{n(n-1)}$ onde $1 / \sqrt{n(n-1)}$ é o raio da maior esfera contida em $e^t x = 1$); o mínimo de uma função linear sobre uma esfera, pode ser encontrado partindo-se do centro da esfera e movendo-se na direção contrária ao gradiente até a superfície da esfera [65]; o passo A0.3

traduz este procedimento. □

Karmarkar [53] apresenta uma demonstração algébrica.

Corolário: O ponto x^{k+1} obtido no passo A0.4 é factível de (P0).

Lema 2 (Lema 2 [11], Lema 4 [67])

$$cD^k x' \leq n^{-1}(1 - \beta\frac{1}{n-1})cx^k$$

Demonstração: Uma vez que o valor ótimo de (P0) é assumido 0, existe um $\bar{x} \geq 0$ tal que $AD^k\bar{x} = 0$, $e^t\bar{x} = 1$ e $cD^k\bar{x} = 0$ ($\bar{x} = \frac{(D^k)^{-1}x^*}{e^t(D^k)^{-1}x^*}$).

A norma máxima:

$$\text{Max } \|x - \frac{1}{n}e\|^2$$

$$\text{sa } e^t x = 1$$

$$x \geq 0$$

é dada pelo ponto $x_i = 1$, $x_{j\neq i} = 0 \ \forall\ i$, logo:

$$\|x - \frac{1}{n}e\|^2 \leq (1 - 1 / n)^2 + \sum_{i=1}^{n-1} (1 / n)^2$$

$$\|x - \frac{1}{n}e\|^2 \leq (n - 1) / n$$

portanto, a esfera de menor raio que contém $e^t x = 1$, $x \geq 0$ é $R = \sqrt{\frac{n-1}{n}}$. Logo o ponto $x_r = (1 - \beta\frac{r}{R})\frac{1}{n}e + \beta\frac{r}{R}\bar{x}$ está contido na esfera $B(\frac{1}{n}e, \beta r)$ (pois $\bar{x}$ está contido em $e^t x = 1$; $x \geq 0$). Portanto:

$$cD^k x' \leq cD^k x_r$$

pois x' minimiza $B(\frac{1}{n}e, \beta r) \cap \Omega^k$.

$$cD^k x' \leq (1 - \beta\frac{r}{R})cD^k\frac{1}{n}e + \beta\frac{r}{R}cD^k x$$

substituindo $\frac{r}{R}$ por $\frac{1}{n-1}$ vem:

$$cD^k x' \leq (1 - \beta\frac{1}{n-1})cD^k\frac{1}{n}e + \beta\frac{1}{n-1}cD^k x$$

$$cD^k x' \leq (1 - \beta\frac{1}{n-1})\frac{1}{n}\, cD^k e = n^{-1}(1 - \beta\frac{1}{n-1})cx^k \quad \square$$

Lema 3 (Lema 4 [11], Lema 5 [67])

Se $\|x - \frac{1}{n}e\| = \beta(n(n-1))^{-1/2}$ e $e^t x = 1$ então $\prod_{i=1}^{n} x_i \geq n^{-n}(1 + \beta\frac{1}{n-1})^{n-1}(1 - \beta)$.

Blair [11] apresenta uma demonstração algébrica, e Padberg [67] demonstra utilizando multiplicadores de Lagrange. Veremos aqui outros argumentos. O ponto ótimo deste problema (Min $\prod_{i=1}^{n} x_i$) é dado pela combinação convexa $x^* = (1 - \beta)\frac{1}{n}e + \beta x(i)$ onde $x(i)$ é a interseção de $e^t x = 1$ com $\|x - \frac{1}{n}e\|$ no hiperplano $x(i)_i = 0$ [04]. Logo $x(i)_{j\neq i} = \frac{1}{n-1}$. Portanto:

$$x_i^* = \frac{1-\beta}{n};\quad x_{j\neq i}^* = \frac{1-\beta}{n} + \beta\frac{1}{n-1} = \frac{(n-1)(1-\beta) + n\beta}{n(n-1)} = n^{-1}(1 + \beta\frac{1}{n-1})$$

Portanto:

$$\prod_{i=1}^{n} x_i^* = n^{-n}(1 + \beta\frac{1}{n-1})^{n-1}(1 - \beta) \leq \prod_{i=1}^{n} x_i \quad \square$$

Lema 4 (Teorema 5 [11])

$$f(x^{k+1}) \leq (1 - \beta\frac{1}{n-1})^n(1 + \beta\frac{1}{n-1})^{1-n}(1 - \beta)^{-1} f(x^k)$$

onde $f(x) = (cx)^n / \prod_{i=1}^{n} x_i$

Observação: Karmarkar [53] utiliza a função potencial $h(x) = \ln(f(x))$. A função utilizada por Blair [11] simplifica as demonstrações.

No passo A0.4, temos: $x^{k+1} = D^k x' / e^t D^k x'$, portanto:

$$f(x^{k+1}) = (cD^k x' / e^t D^k x')^n / \prod_{i=1}^{n} (D^k x' / e^t D^k x')$$

$$f(x^{k+1}) = (cD^k x')^n / \prod_{i=1}^{n} (D^k x') = f(D^k x')$$

pelo Lema 2, o numerador é limitado por:

$$(cD^k x')^n \leq n^{-n}(1 - \beta\frac{1}{n-1})^n (cx^k)^n$$

o denominador pode ser escrito da seguinte forma:

$$\prod_{i=1}^{n} (D^k x') = \prod_{i=1}^{n} x_i^k \prod_{i=1}^{n} x_i'$$

e, pelo Lema 3:

$$\prod_{i=1}^{n} x^k \prod_{i=1}^{n} x' \geq \prod_{i=1}^{n} x^k \; n^{-n}(1 + \beta\frac{1}{n-1})^{n-1}(1 - \beta)$$

portanto:

$$f(x^{k+1}) \leq (1 - \beta\frac{1}{n-1})^n (cx^k)^n / \prod_{i=1}^{n} x^k (1 + \beta\frac{1}{n-1})^{n-1}(1 - \beta)$$

$$f(x^{k+1}) \leq (1 - \beta\frac{1}{n-1})^n (1 + \beta\frac{1}{n-1})^{1-n}(1 - \beta)^{-1} f(x^k) \quad \square$$

Lema 5:

$$\frac{cx^k}{cx^0} \leq (g(\beta,n))^k, \text{ onde}$$

$$g(\beta,n) = (1 - \beta\frac{1}{n-1})(1 + \beta\frac{1}{n-1})^{-1}[(1 + \beta\frac{1}{n-1})/(1 - \beta)]^{1/n}$$

Demonstração:

Do lema 4 obtemos:

$$f(x^{k+1})^{1/n} \leq g(\beta,n)\ f(x^{k})^{1/n}$$

$$[f(x^{k+1}) / f(x^{k})]^{1/n} \leq g(\beta,n)$$

$$[f(x^{k}) / f(x^{0})]^{1/n} \leq (g(\beta,n))^{k}$$

ou seja:

$$\frac{(cx^{k})\ \left[\prod_{i=1}^{n} x_i^{0}\right]^{\frac{1}{n}}}{\left[\prod_{i=1}^{n} x_i^{k}\right]^{\frac{1}{n}}\ (cx^{0})} \leq (g(\beta,n))^{k}$$

$$\frac{cx^{k}}{cx^{0}} \leq n \left[\prod_{i=1}^{n} x_i^{k}\right]^{\frac{1}{n}} (g(\beta,n))^{k}$$

Utilizando a relação entre as médias aritmética e geométrica,

$$\left[\prod_{i=1}^{n} x_i\right]^{\frac{1}{n}} \leq \frac{1}{n}\sum_{i=1}^{n} x_i \text{ temos:}$$

$$\frac{cx^{k}}{cx^{0}} \leq n\ \frac{1}{n}\sum_{i=1}^{n} x_i^{k}\ (g(\beta,n))^{k}$$

$$\frac{cx^{k}}{cx^{0}} \leq (g(\beta,n))^{k}\ \square$$

Teorema 1 (Teorema 1 [53], Teorema único [67])

Em O(np) passos, o algoritmo A0 encontra um ponto x factível tal que:

$$\frac{cx^{k}}{cx^{0}} \leq 2^{-p}$$

Para obter uma estimativa de $g(\beta,n)$, utilizaremos alguns resultados obtidos da série de Taylor

(para $|y| < 1$):

$$\ln \frac{1-y}{1+y} \leq -2y, \text{ portanto (para } y = \frac{\beta}{n-1}),$$

$$\ln \frac{1 - \frac{\beta}{n-1}}{1 + \frac{\beta}{n-1}} \leq \frac{-2\beta}{n-1} \qquad \text{(i)}$$

e da relação $e^y \geq 1 + y$, temos (para $y = \frac{\beta}{n-1}$),

$$\frac{\beta}{n-1} \geq \ln (1 + \frac{\beta}{n-1}) \qquad \text{(ii)}$$

e para $y = -\beta$ temos:

$$\beta + \ln(1 - \beta) \leq 0 \qquad \text{(iii)}$$

tomando o logaritmo de $g(\beta,n)$ temos:

$$\ln(g(\beta,n)) = \ln \frac{1 - \frac{\beta}{n-1}}{1 + \frac{\beta}{n-1}} + \frac{1}{n}\ln (1 + \frac{\beta}{n-1}) - \frac{1}{n}\ln(1 - \beta)$$

usando (i) e (ii) vem:

$$\ln(g(\beta,n) \leq - \frac{\beta}{n-1} - \frac{\beta}{n-1} + - \frac{\beta}{n(n-1)} - \frac{1}{n}\ln(1 - \beta)$$

$$\ln(g(\beta,n) \leq - \frac{\beta}{n-1} - \frac{1}{n}(\beta + \ln(1 - \beta))$$

usando (iii) temos:

$$\ln(g(\beta,n) \leq - \frac{2\beta}{n-1} - \frac{1}{n-1}\ln(1 - \beta)$$

$$k \ln(g(\beta,n) \leq - \frac{k}{n-1} (2\beta + \ln(1 - \beta))$$

para $k \geq \frac{(n - 1)\ p\ \ln(2)}{2\beta + \ln(1 - \beta)}$, ou seja $k \simeq O(np)$ temos:

$$k \ln(g(\beta,n) \leq -p \ln(2)$$

$$(g(\beta,n))^k \leq 2^{-p}$$

pelo Lema 5 temos:

$$\frac{cx^k}{cx^o} \leq 2^{-p}\square$$

Note que $2\beta + \ln(1 - \beta)$ deve ser estritamente

positivo ums vez que p deve ser positivo. Os válidos valores de β estão no intervalo: $0 < \beta < 0{,}7968\ldots$ [67].

Do Teorema 1, temos:

$$cx^k \leq 2^{-p} cx^0$$

No item 2.14.2 vimos que $cx \leq 2^L \ \forall\ x$ factível, logo:

$$cx^k \leq 2^{-p}\ 2^L$$

para $p = 2L$ temos:

$$cx^k \leq 2^{-L}$$

Portanto, em $k \simeq O(nL)$ iterações, no pior caso, encontramos a solução ótima. Como cada iteração tem complexidade polinomial (isto será visto em detalhe no capítulo 3), o algoritmo A0 é polinomial no pior caso.

2.14.4 Outros Métodos Polinomiais

Após a divulgação do trabalho de Karmarkar [53] surgiram outros métodos baseados nas mesmas idéias. Demonstrou-se que esses métodos têm convergência polinomial no pior caso.

Renegar [70] propôs outro algoritmo baseado no método de Newton e na idéia de trajetória central vista no item 2.11. Ele demonstra que este método converge em $O(\sqrt{n+m}L)$ iterações no pior caso.

Baseado na função barreira, pode-se chegar a algoritmos com complexidade ainda menor que o método projetivo [39], $O(\sqrt{n}\ L)$ iterações no pior caso, escolhendo valores apropriados para o parâmetro de barreira a cada iteração. Estes métodos têm a vantagem de não necessitar que o valor da função objetivo no ponto ótimo seja nulo ou de forma equivalente, que o valor ótimo seja conhecido. Porém, tem a desvantagem que, para manter a complexidade teórica, deve-se efetuar passos pequenos, o que aumenta o número médio de iterações [37]. Recentemente, Gonzaga [88,89] obteve a mesma complexidade realizando passos longos para as funções barreira e potencial.

Os métodos baseados na função potencial logarítmica [42,44] (que resolvem o problema em O(nL) iterações) podem realizar passos maiores, sem perder suas propriedades teóricas [37]. Estes algoritmos, assim como o projetivo, necessitam conhecimento prévio do valor ótimo da função objetivo, ou uma estimativa a cada iteração.

O algoritmo primal-dual proposto por Kojima e outros [57] tem complexidade da ordem de nL iterações no pior caso. Monteiro e Adler [90,91] chegam a um resultado melhor $O(\sqrt{n}\ L)$, para um método primal-dual com função

objetivo linear ou quadrática.

Não se conhece até o momento um limitante superior para o número de iterações do algoritmo afim estudado nos itens anteriores. Por outro lado, ele tem as vantagens de não necessitar conhecimento prévio do valor ótimo da função objetivo, não necessitar dos métodos iterativos para a busca unidimensional e, no caso do dual afim, não apresenta problemas numéricos por trabalhar com restrições de desigualdade. Por outro lado, os testes realizados [01,62] até agora parecem indicar que este método tem convergência média próxima dos outros métodos. Recentemente Barnes e outros [07] mostraram que o algoritmo afim com centragem tem complexidade polinomial no pior caso.

Capítulo 3

Conceitos Teóricos Para Implementação

3.1 Introdução

3.1.1 Objetivo

Vimos no capítulo 2 métodos de ponto interior que procuram encontrar a solução de problemas lineares em um número razoavelmente pequeno de iterações. No entanto, uma iteração de um algoritmo de ponto interior tem complexidade (requer um esforço computacional) maior que uma iteração do simplex, o método tradicional para solução de problemas lineares. Ou seja, na realização de cada iteração os métodos de ponto interior são mais lentos do que o simplex.

Não entrando em detalhes, uma iteração do simplex envolve uma atualização de "rank 1" de uma matriz (pivoteamento), enquanto que uma iteração de ponto interior envolve a resolução de um sistema linear.

Em linhas gerais, no capítulo 2 nos preocupamos em desenvolver métodos que diminuíssem o número total de iterações, mesmo ao custo de encarecê-las (preservando sempre que possível a complexidade polinomial).

Neste capítulo nossa preocupação será a diminuição do custo de cada iteração, mesmo que isto possa aumentar (um pouco) o número total delas.

3.1.2 Sistema Linear

A passagem mais cara, em termos de esforço computacional, dos métodos vistos no capítulo anterior, envolve a resolução do seguinte sistema linear:

$$(AD_k^* A^t)p^k = q^k$$

Observações:

- k refere-se ao número da iteração;
- Conforme o método utilizado e o tipo de problema a ser resolvido, pode ser necessário calcular q^k antes de resolver o sistema, de qualquer forma, o cálculo de q^k é pelo menos uma ordem menor que o cálculo de $AD_k^* A^t$;
- A matriz $AD_k^* A^t$ é simétrica definida positiva e a cada iteração somente a matriz diagonal D_k^* se altera.

A resolução de sistemas lineares aparece em áreas clássicas da matemática aplicada, não sendo portanto um problema novo. Existem, no entanto, muitos avanços recentes em relação a eficiência na resolução deste tipo de problema. A maior parte das técnicas que serão vistas neste capítulo, tem aplicação na resolução de sistemas lineares

genéricos. Outras levarão em consideração o fato que o sistema a ser resolvido faz parte de uma iteração de um método de ponto interior para programação linear.

3.2 Esparsidade

O aspecto que leva ao maior ganho no tempo de cada iteração (além da economia em memória) é a consideração da estrutura esparsa das matrizes A e $AD_k^* A^t$. Em problemas de pequeno porte (com cerca de 20 linhas), chega-se a diminuir o tempo de cada iteração em mais de 10 vezes, conforme o grau de esparsidade das matrizes. Problemas de médio porte podem chegar facilmente a ganhos de 100 vezes.

Quase todas as técnicas apresentadas neste capítulo tem em vista o melhor aproveitamento da estrutura esparsa das matrizes envolvidas. Em resumo, todas as técnicas buscam um único objetivo: realizar o menor número possível de operações por iteração.

3.3 Decomposição

3.3.1 Motivação

A resolução do sistema através do cálculo explícito da inversa está fora de cogitação, pois, mesmo que

$AD_k^* A^t$ seja esparsa, o mesmo não ocorre necessariamente com a sua inversa. Portanto métodos que não calculam a inversa da matriz são preferíveis. Uma das formas mais populares de se resolver um sistema linear é a decomposição da matriz.

Uma vez que a matriz $AD_k^* A^t$ é simétrica positiva definida, o sistema pode ser resolvido com a decomposição de Cholesky (LEL^t ou LL^t), onde L é triangular inferior e E é diagonal. Esta decomposição utiliza aproximadamente metade do esforço computacional da decomposição LU (veja que sendo LEU única $L^t = U$). Além do mais, ela é intrinsicamente estável quando a matriz decomposta é positiva definida, o que dispensa avaliações numéricas na escolha do pivô [28].

A segunda delas (LEL^t) é preferível, em princípio, pois ao contrário da LL^t não necessita da operação raiz quadrada (que é cara em algumas máquinas) para o cálculo da matriz L. Isto porque os elementos das diagonais de L e U (L^t neste caso) são unitários na decomposição LEU [75], o que traz ganhos também na resolução dos sistemas triangulares, a fase seguinte após a decomposição.

Obtida a decomposição, o sistema fica da seguinte forma:

$$LEL^{t}p = q$$

este sistema pode ser facilmente resolvido por substituição em dois passos:

$$Lr = q$$

$$L^{t}p = E^{-1}r$$

Outra forma de buscar a solução do sistema no método primal, é aplicar a decomposição QR [75,38] no sistema:

$$(AD^{*}A^{t})y^{k} = AD^{*}c$$

Tomlin [79] descreve uma implementação desta forma e apresenta resultados comparativos do método projetivo com o simplex. Ele conclui que o método projetivo tem dificuldades de superar o simplex devido ao elevado esforço computacional de cada iteração.

Ainda é possível resolver o sistema através da fatoração da inversa obtendo $(AD^{*}A^{t})^{-1} = UL$, este método é conhecido como forma produto da inversa [85].

A resolução do sistema via "loop-free code" [45] é provalvemente o método mais rápido de implementar a decomposição, principalmente no caso do sistema ser resolvido várias vezes com a mesma estrutura. Por outro lado, o tamanho do código gerado inviabiliza a sua utilização na maioria dos ambientes computacionais

existentes. Em [86] é feita uma implementação baseada em geração simbólica que é um meio termo entre o código com "loop" e o "loop-free code".

3.3.2 Decomposição de Cholesky

A decomposição pode ser feita de várias maneiras, conforme os elementos da matriz L são calculados. Vamos estudar a decomposição que calcula os elementos da matriz L por coluna. Em [30] é feito um estudo detalhado da implementação da decomposição na versão linha.

Veremos a seguir o algoritmo de decomposição por coluna $B = LEL^t$ com atualização por coluna. Este algoritmo é uma especialização do procedimento apresentado em [75] para o caso não simétrico. Eles apresentam vantagem computacional sobre os algoritmos propostos recentemente em [38], embora a ordem de complexidade seja a mesma.

Para a apresentação do algoritmo e o cálculo da complexidade, usaremos a seguinte notação [72,45]:

Seja $B_{(m \times m)}$ uma matriz qualquer, então:

B_i^j = Elemento da linha i, coluna j de B;

B^j = j-ésima coluna de B;

B_i = i-ésima linha de B;

$$\hat{B}_i^j = \begin{cases} 0 \text{ se } B_i^j = 0; \\ 1 \text{ caso contrário;} \end{cases}$$

$\hat{B}^j = \sum_{i=1} \hat{B}_i^j$ = Número de elementos não nulos da j-ésima coluna de B;

$\hat{B}_i = \sum_{j=1} \hat{B}_i^j$ = Número de elementos não nulos da i-ésima linha de B;

$\hat{B} = \sum_{i=1} \sum_{j=1} \hat{B}_i^j$ Número de elementos não nulos da matriz.

Decomposição LEL^t por coluna, atualização por coluna

A8.

```
Para i=1 até m
    E_i^i ← B_i^i                                   A8.1
    Para j=i+1 até m
        L_j^i ← B_j^i / B_i^i                       A8.2
        Para k=j até m
            B_k^j ← B_k^j - L_j^i B_k^i             A8.3
```

Observações:

- A matriz B é alterada durante o processo.

- O algoritmo pode ser facilmente modificado para que os elementos de LEL^t sejam guardados na mesma posição de B.

3.3.3 Complexidade da Decomposição para Matrizes Cheias

Para o cálculo do número de operações, serão usadas as seguintes fórmulas [38,76]:

$$\sum_{k=1}^{n} k^2 = \frac{(n+1)n(2n+1)}{6} \qquad (1)$$

$$\sum_{k=i+1}^{n} k = \frac{(n-i)(n+i+1)}{2} \qquad (2)$$

O somatório (2) pode ser deduzido do resultado:

$$\sum_{l=1}^{n} l = \frac{n(n+1)}{2} \quad \text{definindo } l = k-i.$$

No desenvolvimento a seguir, serão consideradas somente as operações de multiplicação e divisão. A partir dos resultados finais, é fácil obter a complexidade, levando em conta as outras operações.

Somente os passos (A8.2) e (A8.3) dos algoritmos têm operações de multiplicação ou divisão. Veremos primeiramente o número de operações do passo (A8.2), que é comum às duas versões:

Seja Op(A8.2) o número de multiplicações e divisões do passo A8.2. Temos:

$$Op(A8.2) = \sum_{i=1}^{m} \sum_{j=i+1}^{m} 1 = \sum_{i=1}^{m} (m-i)$$

Usando o resultado (2), temos:

$$Op(A8.2) = m(m-1) - \frac{(m-1)m}{2} = \frac{m(m-1)}{2} = \frac{m^2 - m}{2}$$

Para o passo (A8.3), temos:

$$Op(A8.3) = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{k=j}^{m} 1 = \sum_{i=1}^{m} \sum_{j=i+1}^{m} (m-j+1)$$

$$Op(A8.3) = \sum_{i=1}^{m} \left[(m+1)(m-i) - \frac{(m-i)(m+i+1)}{2} \right]$$

$$Op(A8.3) = \sum_{i=1}^{m} \frac{(m-i)(m-i+1)}{2}$$

Desenvolvendo a fórmula acima temos:

$$Op(A8.3) = \sum_{i=1}^{m} \frac{(m-i)(m-i+1)}{2} = \frac{1}{2} \sum_{i=1}^{m} \left[m(m+1) - (2m+1)i + i^2 \right]$$

Aplicando as fórmulas (1) e (2), temos:

$$Op(A8.3) = \frac{1}{2} \left[m(m+1)m - \frac{(2m+1)(m+1)m}{2} + \frac{(m+1)m(2m+1)}{6} \right]$$

$$Op(A8.3) = \frac{m^2 + m}{2} \left[\frac{6m - 6m - 3 + 2m + 1}{6} \right] = \frac{(m^2+m)(m-1)}{6} = \frac{m^3 - m}{6}$$

Assim, considerando os passos (A8.2) e (A8.3) verificamos que o número de operações de multiplicação e

divisão do algoritmo (Op(A8) = Op(A8.2) + Op(A8.3)) é dado por:

$$Op(A8) = \frac{m^3 - m}{6} + \frac{m^2 - m}{2}$$

3.3.4 Complexidade da Decomposição para Matrizes Esparsas

Na decomposição de matrizes esparsas, podem surgir elementos não nulos em posições onde existiam elementos nulos na matriz original (fill-in). A análise será feita considerando-se estes elementos.

Para o passo (A8.2), temos:

$$Op(A8.2) = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \hat{L}_j^i = \sum_{i=1}^{m} (\hat{L}^i - 1) = \sum_{i=1}^{m} \hat{L}^i - \sum_{i=1}^{m} 1 = \hat{L} - m$$

Para o passo (A8.3), temos:

$$Op(A8.3) = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{k=j}^{m} \hat{L}_j^i \, \hat{L}_k^i \, \hat{L}_k^j$$

Observando que se L_j^i e L_k^i forem diferentes de zero, então $L_k^j \neq 0$, ou seja, $\hat{L}_j^i = \hat{L}_k^i = 1$ implica $\hat{L}_k^j = 1$. Portanto, o número de vezes que o passo A8.3 é executado não é função de $\hat{L}_k^j$, logo:

$$Op(A8.3) = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{k=j}^{m} \hat{L}_j^i \, \hat{L}_k^i = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \hat{L}_j^i \sum_{k=j}^{m} \hat{L}_k^i$$

$$Op(A8.3) = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \ell_j^i \ell_J^i$$

onde J = (j,..,m).

O termo $\ell_j^i \ell_J^i$ é a combinação de cada elemento ℓ_j^i com ele próprio e com todos elementos não nulos abaixo dele, portanto, o somatório $\sum_{j=i+1}^{m} \ell_j^i \ell_J^i$ vale 1 quando j é o índice do último elemento não nulo da coluna i, vale 2 quando j é o índice do penúltimo elemento e assim sucessivamente. Como o somatório é feito para todos elementos da coluna i exceto o elemento da diagonal, temos:

$$\sum_{j=i+1}^{m} \ell_j^i \ell_J^i = \sum_{l=1}^{\ell^i-1} l = \frac{(\ell^i-1)(\ell^i-1+1)}{2}$$

portanto:

$$Op(A8.3) = \sum_{i=1}^{m} \frac{(\ell^i-1)\ell^i}{2}$$

Ou seja, para cada elemento não nulo de uma coluna, exceto na diagonal, atualize a coluna correspondente.

Finalmente, podemos obter o total de operações.

$$Op(A8) = Op(A8.2) + Op(A8.3) = \sum_{i=1}^{m} \left[\frac{\ell^i(\ell^i-1)}{2}\right] + \ell - m$$

$$Op(A8) = \sum_{i=1}^{m} \left[\frac{\ell^i (\ell^i + 1)}{2} \right] - m$$

Pode-se verificar que para matrizes cheias ($\ell^i = m - i + 1$), a substituição destes valores na fórmula acima obtém o resultado deduzido no item anterior 3.3.3. para Op(A8). Também é fácil verificar que a resolução dos sistemas triangulares realiza m^2 operações para matrizes cheias e $2\ell - m$ operações para matrizes esparsas.

3.3.5 Complexidade no Cálculo de AD^*A^t

Antes de resolver o sistema, é necessário obter a matriz $B = AD^*A^t$. Neste item veremos qual o esforço computacional que seu cálculo exige.

A9.

```
Para i=1 até m
    Para j=1 até i
        B_i^j ← 0                                   A9.1
        Para k=1 até n
            B_i^j ← B_i^j + A_i^k A_j^k (D_k^k)*    A9.2
        B_j^i ← B_i^j                               A9.3
```

Supondo que $(D_k^k)^*$ já esteja calculado, a quantidade de operações realizadas é duas vezes o número de execuções do passo (A9.2).

Matriz cheia

$$Op(A9) = 2\sum_{i=1}^{m}\sum_{j=1}^{i}\sum_{k=1}^{n}1 = 2\sum_{i=1}^{m}\sum_{j=1}^{i}n = 2n\sum_{i=1}^{m}i$$

Do resultado (2) vem:

$$Op(A9) = \frac{2n(m+1)m}{2} = nm(m+1)$$

Matriz esparsa

$$Op(A9) = 2\sum_{i=1}^{m}\sum_{j=1}^{i}\sum_{k=1}^{n}\hat{A}_i^k\hat{A}_j^k = 2\sum_{k=1}^{n}\sum_{i=1}^{m}\sum_{j=1}^{i}\hat{A}_i^k\hat{A}_j^k$$

$$Op(A9) = 2\sum_{k=1}^{n}\sum_{i=1}^{m}\hat{A}_i^k\hat{A}_I^k$$

onde $I = (1,..,i)$

Analogamente ao caso anterior, $\hat{A}_i^k\hat{A}_I^k$ é a combinação do elemento $\hat{A}_i^k$ com ele mesmo e com todos os elementos não nulos acima dele. Portanto o somatório pode ser reescrito da seguinte forma:

$$\sum_{i=1}^{m}\hat{A}_i^k\hat{A}_I^k = \sum_{l=1}^{\hat{A}^k}l$$

$$Op(A9) = 2\sum_{k=1}^{n}\sum_{l=1}^{\hat{A}^k}1 = 2\sum_{k=1}^{n}\frac{\hat{A}^k(\hat{A}^k+1)}{2} = \sum_{k=1}^{n}\hat{A}^k(\hat{A}^k+1)$$

Adler e outros [86] e Monma & Morton [62] armazenam o produto $A_i^k A_j^k$ que não varia com a iteração e, desta forma, conseguem metade do esforço no cálculo de AD^*A^t, desconsiderando outras operações e a obtenção de $A_i^k A_j^k$ que é feita apenas um vez. No entanto, esta forma pode criar problemas de armazenamento pois necessita $\sum_{k=1}^{n} \frac{\hat{A}^k(\hat{A}^k+1)}{2}$ posições de memória para armazenar cada produto individualmente.

3.3.6 Complexidade do Método Projetivo

No capítulo 2 vimos a demonstração que o método projetivo converge em O(nL) iterações, onde L é o tamanho da entrada em bits. Vimos no item anterior que cada iteração é da ordem de m^2n operações aritméticas, uma vez que $m \leq n$, podemos escrever $O(n^3)$ no pior caso, portanto, a complexidade do método projetivo no pior caso é $O(n^4L)$ operações aritméticas.

Segundo [02], cada operação aritmética pode ser resolvida em um tempo de ordem $O(L \cdot \log(L) \cdot \log(\log(L)))$ em uma máquina de Turing.

No próximo item veremos que este valor pode ser reduzido utilizando atualização de "rank 1". Veja que os resultados obtidos neste item para a complexidade da

iteração vale para todos os métodos de ponto interior estudados, pois eles resolvem um sistema da mesma forma.

3.4 Atualização das Matrizes do Sistema

A cada iteração, apenas a matriz diagonal do produto $B = AD^*A^t$ se modifica. Karmarkar [53] demonstrou para o método projetivo que se algum elemento de D que foi modificado "significativamente" não for alterado, a convergência do método não fica comprometida; além disto, demonstrou que o número de elementos que variam muito é, na média, pequeno ($O(\sqrt{n})$ em m passos).

Usando o resultado acima, o produto AD^*A^t pode ser calculado mais facilmente:

$$AD^*A^t = A\underline{D}^*A^t + \sum_{i \in \phi} A^i((D_i^i)^* - (\underline{D}_i^i)^*)(A^i)^t$$

Onde ϕ é o conjunto dos elementos da diagonal que variaram significativamente na última iteração e $\underline{D}^*$ é a matriz diagonal da iteração anterior.

3.4.2 Atualização da Decomposição

Shanno [74] utilizou esta idéia para atualizar as matrizes L e E diretamente, como veremos a seguir.

Seja ϕ o conjunto de elementos que modificaram "significativamente". Então a nova decomposição é dada por:

$$LEL^t = \underline{L}\underline{E}\underline{L}^t + \sum_{i \in \phi} A^i \left[(D_i^i)^2 - (\underline{D}_i^i)^2 \right] (A^i)^t$$

Fletcher e Powell [31] apresentaram um algoritmo de atualização da decomposição para a forma:

$$LEL^t = \underline{L}\underline{E}\underline{L}^t + \sigma z z^t$$

Este algoritmo pode ser aplicado $|\phi|$ vezes (cardinalidade do conjunto ϕ) para se obter a nova decomposição. Eles também levam em conta o problema de instabilidade numérica na atualização e o fato da matriz B ser definida positiva.

Uma vez que na prática $|\phi|$ varia muito para os métodos primal e dual afim, podendo inclusive ser maior que m por várias iterações, optamos pela atualização proposta em [51] e [49], que é mais rápida e aplicável a uma matriz genérica, embora não leve em consideração a instabilidade numérica (isto não traz inconvenientes para o problema em estudo).

Tanto a atualização de [31] como a de [51] foram adaptadas da proposta em [10], que é mais geral, podendo inclusive atualizar a decomposição para todo o

conjunto ϕ simultaneamente utilizando a partição $A^{\phi}D^{*\phi}(A^{\phi})^{t}$.

Vemos a seguir o algoritmo de atualização da matriz de [51] e [49].

Atualização da decomposição [51], [49]

A10.

Para i=1 até m

$\alpha \leftarrow E_i^i$ A10.1

$E_i^i \leftarrow \alpha + \sigma z_i^2$ A10.2

  Para j=i+1 até m

    $z_j \leftarrow z_j - z_i L_j^i$ A10.3

    $L_j^i \leftarrow L_j^i + \sigma z_i z_j / E_i^i$ A10.4

$\sigma \leftarrow \sigma \alpha / E_i^i$ A10.5

Neste algoritmo z representa a coluna que está sendo atualizada.

Fizemos uma pequena alteração no algoritmo A10 para reduzir o número total de operações. A seguir vemos o algoritmo alterado.

Atualização da decomposição com a alteração

A11.

Para i=1 até m

$\alpha \leftarrow E^i_i$ A11.1

$E^i_i \leftarrow \alpha + \sigma z^z_i$ A11.2

Se i = m

Fim

$\beta \leftarrow \sigma / E^i_i$ A11.3

$\gamma \leftarrow \beta z_i$ A11.4

$\sigma \leftarrow \alpha\beta$ A11.5

Para j=i+1 até m

$z_j \leftarrow z_j - z_i L^i_j$ A11.6

$L^i_j \leftarrow L^i_j + z_j\gamma$ A11.7

3.4.3 Complexidade da Atualização da Decomposição

Matriz cheia

Para o passo (A11.2), temos:

$$Op(A11.2) = 2\sum_{i=1}^{m} 1 = 2m$$

Para os passos (A11.3), (A11.4) e (A11.5), temos:

$$Op(A11.3) + Op(A11.4) + Op(A11.5) = 3\sum_{i=1}^{m-1} 1 = 3(m-1)$$

Para os passos (A11.6) e (A11.7), temos (utilizando o resultado Op(A8.2) do item 3.3.3):

$$Op(A11.6) + Op(A11.7) = 2\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} 1 = 2\sum_{i=1}^{m}\sum_{j=i+1}^{m} 1 = m^2 - m$$

O número total de multiplicações e divisões é dado por:

$$Op(A11) = Op(A11.3)+Op(A11.4)+Op(A11.5)+Op(A11.6)+Op(A11.7)$$

$$Op(A11) = 2m + 3(m-1) + m^2 - m = m^2 + 4m - 3$$

Matriz esparsa

Considerando o vetor z não esparso, o cálculo para os passos (A11.2), (A11.3), (A11.4) e (A11.5) não se alteram.

Para os passos (A11.6) e (A11.7), temos (utilizando o resultado Op(A8.2) do item 3.3.4):

$$Op(A11.6) + Op(A11.7) = 2\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} \ell_j^i = 2\sum_{i=1}^{m}\sum_{j=i+1}^{m} \ell_j^i = 2(\ell - m)$$

Total de operações:

$$Op(A11) = 2m + 3(m-1) + 2(\ell - m) = 2\ell + 3(m-1)$$

Comentário: Não está sendo levada em conta, a esparsidade do

vetor z no algoritmo. Pode-se verificar que os primeiros k-1 elementos tal que $z_{k-1} = 0$ não alteram a decomposição.

Levando isto em consideração o número de operações dos passos (A11.6) e (A11.7) fica:

$$Op(A11.6) + Op(A11.7) = 2\sum_{i=k}^{m}\sum_{j=i+1}^{m} \ell_j^i = 2\sum_{i=k}^{m} \ell^i - (m - k + 1)$$

O número de operações no passo A11.2 fica:

$$Op(A11.2) = 2\sum_{i=k}^{m} 1 = 2(m - k + 1)$$

Os passos (A11.3), (A11.4) e (A11.5) ficam:

$$Op(A11.3) + Op(A11.4) + Op(A11.5) = 3\sum_{i=k}^{m-1} 1 = 3(m - k)$$

Finalmente, o total de operações é dado por:

$$Op(A11) = 2\sum_{i=k}^{m} \ell^i - 3(m - k)$$

Note que para os métodos de ponto interior, k representa a linha do primeiro elemento não nulo da coluna que está sendo atualizada.

A complexidade da iteração no pior caso com a atualização da decomposição fica, portanto, reduzida a $m^2\sqrt{n} \leq n^{2.5}$, logo o método projetivo tem uma complexidade no

pior caso de $O(n^{3.5}L)$ operações aritméticas.

3.4.4 Escolha entre Decomposição e Atualização

A cada iteração, é possível escolher qual o caminho que faz menos operações, uma nova decomposição ou $|\phi|$ atualizações. Uma nova decomposição necessita do produto $AD^{*}A^{t}$.

O ponto de corte

$$|\phi|_c \, O(\text{atualização}) = O(\text{multiplicação}) + O(\text{decomposição})$$

é dado por:

Matriz cheia

Usando os resultados do item 3.4.3 para matrizes cheias temos:

$$|\phi|_c = \frac{\frac{nm(m+1)}{2} + \frac{m^3 - m}{6} + \frac{m^2 - m}{2}}{(m^2 + 4m - 3)}$$

Para se ter uma idéia qualitativa, considere m e n grandes, então:

$$|\phi|_c = \frac{\frac{nm^2}{2} + \frac{m^3}{6}}{m^2} \simeq O(n)$$

Ou seja, para matrizes cheias suficientemente grandes, sempre vale a pena fazer-se as atualizações. Esta é a

situação (sem esparsidade) estudada por Shanno [74].

Usando os resultados do item 3.4.3 para matrizes esparsas temos:

$$|\phi|_c = \frac{\sum_{k=1}^{|\phi|} \frac{\hat{A}^k(\hat{A}^k+1)}{2} + \sum_{i=1}^{m} \frac{\hat{L}^i(\hat{L}^i+1)}{2} - m}{2\sum_{i=k(j)}^{m} \hat{L}^i - 3(m - k(j))}$$

Onde k(j) é a linha do primeiro elemento não nulo da coluna j.

Portanto,

Quando $|\phi| > |\phi|_c$, decompõe-se a matriz.

Quando $|\phi| \leq |\phi|_c$, atualiza-se a decomposição.

Veja que $|\phi|_c$ pode variar a cada iteração em função das colunas a serem atualizadas. Isto não ocorre quando a esparsidade das colunas não é considerada.

Note que do ponto de vista de diminuir o número total de operações, sempre vale a pena atualizar AD^*A^t quando for realizada uma nova decomposição. Por outro lado, isto pode produzir problemas numéricos, sendo recomendável um novo cálculo de AD^*A^t periodicamente.

3.4.5 Atualização da Hessiana

Este tipo de abordagem se aplica nos métodos de ponto interior baseados em funções não lineares. Alguns destes foram comentados brevemente no capítulo 2 (item 2.10). A abordagem consiste em calcular a nova direção utilizando uma aproximação da Hessiana da função não linear, a partir da Hessiana da iteração anterior.

Em [35] é sugerida a utilização por algumas iterações da mesma Hessiana, modificando apenas o lado direito da equação q^k que é função de D_k^* (neste caso, o algoritmo utilizado é baseado na função barreira para o problema dual (D) permitindo a manutenção da factibilidade). Esta idéia aproveita a decomposição da iteração anterior.

Em [25] a Hessiana é calculada por uma fórmula de atualização, utilizando diferentes funções (potencial e barriera). Neste caso a factibilidade é mantida. Uma forma similar de atualização é proposta por Martinez [60].

3.5 Reordenamento da Matriz

Já observamos que apenas a matriz diagonal D^* se altera a cada iteração. Logo, a estrutura de AD^*A^t fica preservada durante todo o processo. Este fato permite que se faça um pré-processamento simbólico da matriz, para identificar quais os elementos são estruturalmente nulos, ou seja são iguais a zero em todas as iterações.

É preciso também considerar que o processo de decomposição é capaz de modificar radicalmente a esparsidade estrutural do sistema através da ocorrência de "fill-in" (enchimento), que são elementos estruturalmente nulos em AD^*A^t e não são nulos (necessariamente) em L. Neste caso, uma matriz esparsa pode resultar em uma matriz cheia após a decomposição, o que é indesejável.

O número de "fill-ins" que surge é função da ordem em que estão colocadas as linhas e colunas de AD^*A^t (que por sua vez é função da ordem das linhas de A). Logo, se pudermos encontrar a ordenação que garante o mínimo "fill-in", o número de operações por iteração será menor.

Infelizmente, o problema "minimum global fill-in" é NP-completo [82], o que torna sua solução mais difícil do que o problema original. Entretanto, várias

heurísticas têm sido usadas para reordenar matrizes de forma a preservar, mesmo que parcialmente, a esparsidade durante a fatoração [28]. Duas delas, "minimum degree" e "minimum local fill-in" foram testadas com bons resultados para matrizes originadas de problemas de programação linear [14]. Note que, sendo a matriz definida positiva, o reordenamento pode ser feito levando-se em consideração apenas a estrutura da matriz (sem consideração de aspectos de estabilidade numérica).

O número de operações na decomposição não é função apenas da quantidade de "fill-in" que surge, mas também da sua distribuição nas colunas como foi visto no item 3.3.4. Carvalho [14] apresenta um exemplo simples em que o esforço da decomposição é diferente para duas ordenações que geram o mesmo número de "fill-in" - ele comenta que é dada muita atenção ao número de "fill-in" e nenhuma ao esforço da decomposição. De fato, como vimos no item 3.3.3, a função a minimizar é $\sum_{i=1}^{m} \frac{\hat{L}^i(\hat{L}^i+1)}{2} - m$. Esta função depende do número total de elementos não nulos de L e também da sua distribuição pelas colunas.

Descrevemos a seguir a heurística que utilizamos ("minimum degree"). Em [28] várias outras heuríticas são descritas em detalhe.

O método "minimum degree", para matrizes simétricas, foi proposto por Tinney & Walker [77] (esquema 2). Ele recebe este nome devido a sua interpretação em teoria de grafos. A estratégia consiste em escolher o nó com menor grau no grafo associado à matriz. Isto corresponde a selecionar a linha (e coluna pois a matriz neste caso é simétrica) com menor número de elementos não nulos como a próxima linha do pivô. Após a escolha do pivô, repete-se o processo para a matriz resultante, no esquema 1 [77], sem atualização da matriz resultante e, no esquema 2 com atualização incluindo os "fill-in" que eventualmente surjam. O esquema 1 é mais rápido, porém, em geral produz resultados piores em termos do número de "fill-ins", o que justifica a utilização do esquema 2 [77].

O ponto crítico do reordenamento "minimum degree" é a atualização do grau dos nós resultantes após uma eliminação, pois é preciso levar em conta não só os "fill-in" que surgem nesta eliminação mas também em todas as anteriores. Após a escolha de um pivô (nó), apenas os nós adjacentes a ele antes da sua escolha têm o grau alterado. Deve-se subtrair uma ligação referente ao nó retirado e acrescentar as ligações referentes aos novos "fill-ins". No grafo, um "fill-in" aparece quando dois nós adjacentes ao pivô não são adjacentes entre si.

O reordenamento pode ser feito sem utilizar posições de memória para representar os "fill-ins" utilizando o conceito de clique [28] (clique é um grafo completo). Neste caso, a localização dos "fill-ins" é feita em uma fase subsequente [30].

3.6 Métodos Iterativos

3.6.1 Considerações Iniciais

A utilização de decomposição para resolver o sistema linear é interessante quando a matriz AD^*A^t é esparsa. Em algumas situações, mesmo que a matriz A seja esparsa, é possível que a matriz AD^*A^t não seja, por exemplo, quando a matriz A tem uma coluna completamente cheia, a matriz AD^*A^t será completamente cheia. Em casos como este, a utilização de métodos iterativos de convergência rápida tende a ser mais eficiente que a decomposição. Esta abordagem tem vantagens e desvantagens em relação aos métodos diretos.

Vantagens:

- Os métodos iterativos não geram "fill-ins";
- Não necessitam reordenar a matriz;

Desvantagens:

- Os métodos iterativos geram soluções inexatas podendo comprometer a convergência do método primal ($Ax = b$) e, no método dual, obter uma estimativa ruim para as variáveis primais ($x = D^{*}A^{t}dy$);

- As extensões que necessitam resolver mais de um sistema linear por iteração com a mesma matriz, como a centragem, podem se tornar muito caras.

3.6.2 Gradiente Conjugado Pré-Condicionado

Um dos métodos iterativos mais eficientes para a resolução de um sistema linear é o gradiente conjugado pré-condicionado (GCPC) [38,46,58]. A utilização de um algoritmo de otimização pode ser feita porque encontrar a solução de $Bp = q$ equivale a minimizar a função $\frac{1}{2}p^{t}Bp - p^{t}q$.

A seguir apresentamos o algoritmo GCPC com uma matriz de condicionamento genérica C.

Dados p^{o} um ponto inicial

$e \in_{gc}$ uma tolerância;

faça $k \leftarrow 0$

$r^{o} \leftarrow q - Bp^{o}$ A12.1

$\hat{r}^{o} \leftarrow C^{-1}r^{o}$ A12.2

$dp^{o} \leftarrow \hat{r}^{o}$ A12.3

A12.

Repita

$s \leftarrow Bdp^k$ A12.4

$\alpha \leftarrow (r^k, \hat{r}^k) / (dp^k, s)$ A12.5

$p^{k+1} \leftarrow p^k + \alpha dp^k$ A12.6

$r^{k+1} \leftarrow r^k - \alpha s$ A12.7

$\hat{r}^{k+1} \leftarrow C^{-1} r^{k+1}$ A12.8

SE $(r^{k+1}, \hat{r}^{k+1}) < \in_{gc}$ RETORNE(p^{k+1}) A12.9

$\beta \leftarrow (r^{k+1}, \hat{r}^{k+1}) / (r^k, \hat{r}^k)$ A12.10

$dp^{k+1} \leftarrow \hat{r}^{k+1} + \beta dp^k$ A12.11

$k \leftarrow k + 1$

Observação:

- O passo A12.7 pode ser feito desta forma pois

$$r^k - \alpha s = q - Bp^{k+1}$$

assim, o custo do GCPC é da ordem de km^2 ($k\hat{B}$ com esparsidade) operações aritméticas, considerando que o cálculo de $C^{-1}r$ não exija um esforço maior que m^2, que é o caso das implementações práticas.

Heath [46] apresenta várias formas de pré-condicionamento através da escolha da matriz C. Adler e outros [01] usam uma fatorização de Cholesky incompleta como pré-condicionamento. A idéia deles é a seguinte: seja N o conjunto de colunas de A que contém menos que um certo número de elementos não nulos fixo, e L a decomposição de

Cholesky de $A^N D^2 A^{N^t}$. O sistema linear $Bp = q$ pode ser reescrito da seguinte forma:

$$L^{-1}BL^{-t}L^t p = L^{-1}q$$

ou seja,

$$B'p' = q'$$

onde $B' = L^{-1}BL^{-t}$; $p' = L^t p$; e $q' = L^{-1}q$.

O algoritmo gradiente conjugado ($C = I$) é aplicado ao sistema resultante.

Eisenstat [29] propõe uma implementação eficiente para o GCPC com a matriz $C = (\tilde{E} + L)\tilde{E}^{-1}(\tilde{E} + L)^t$ onde $B = L + E + L^t$; L é estritamente triangular inferior; E e $\tilde{E}$ são diagonais positivas. Ele reescreve o sistema linear da seguinte forma:

$$(\tilde{E} + L)^{-1}B(\tilde{E} + L)^{-t}(\tilde{E} + L)^t p = (\tilde{E} + L)^{-1}q$$

ou seja,

$$B'p' = q'$$

onde B', p' e q' são definidos de forma análoga a de Adler e outros [01] descrita acima.

A aplicação do algoritmo A12. em $Bp = q$ com, $C = (\tilde{E} + L)\tilde{E}^{-1}(\tilde{E} + L)^t$ é equivalente a aplicar o mesmo algoritmo em $B'p' = q'$ com $C = \tilde{E}^{-1}$ e $(\tilde{E} + L)^t p = p'$. Se atualizarmos p no lugar de p' no algoritmo A12, vamos obter o seguinte algoritmo:

A13.

Dados p^0 um ponto incicial

e $\in_{gc}$ uma tolerância;

faça $k \leftarrow 0$

$r'^0 \leftarrow (\tilde{E} + L)^{-1}(q - Bp^0)$ A13.1

$\hat{r}'^0 \leftarrow \tilde{E}r'^0$ A13.2

$dp'^0 \leftarrow \hat{r}'^0$ A13.3

Repita

- $s' \leftarrow B'dp'^k$ A13.4
- $\alpha \leftarrow (r'^k, \hat{r}'^k) / (dp'^k, s')$ A13.5
- $p^{k+1} \leftarrow p^k + \alpha(\tilde{E} + L)^{-t}dp'^k$ A13.6
- $r'^{k+1} \leftarrow r'^k - \alpha s'$ A13.7
- $\hat{r}'^{k+1} \leftarrow \tilde{E}r'^{k+1}$ A13.8
- SE $(r'^{k+1}, \hat{r}'^{k+1}) < \in_{gc}$ RETORNE(p^{K+1}) A13.9
- $\beta \leftarrow (r'^{k+1}, \hat{r}'^{k+1}) / (r'^k, \hat{r}'^k)$ A13.10
- $dp'^{k+1} \leftarrow \hat{r}'^{k+1} + \beta dp'^k$ A13.11
- $k \leftarrow k + 1$

Este algoritmo é eficiente pois $B'dp'$ pode ser calculdado facilmente na forma descrita a seguir:

$$B'dp' = (\tilde{E} + L)^{-1}\left[(\tilde{E} + L) + (\tilde{E} + L)^t - (2\tilde{E} - E)\right](\tilde{E} + L)^{-t}dp'$$

$$B'dp' = (\tilde{E} + L)^{-t}dp' + (\tilde{E} + L)^{-1}\left[dp' - K(\tilde{E} + L)^{-t}dp'\right]$$

onde $K = 2\tilde{E} - E$. Seja $t = (\tilde{E} + L)^{-t}dp'$, então:

$$B' dp' = t + (\tilde{E} + L)^{-1}(dp' - Kt)$$

Portanto o maior esforço deste algoritmo está na resolução de dois sistemas triangulares ($\hat{B}$ + 2m operações). Veja que t pode ser utilizado no passo A13.6 para calcular dp^{k+1}. A utilização de $\tilde{E}$ = I economiza 2m divisões. Outra escolha interessante, utilizada na implementação, é $\tilde{E}$ = E o que leva a K = E.

3.7 Redução da Dimensão do Problema

Uma das formas que tem sido utilizada por diversos autores para reduzir o esforço computacional de cada iteração é a redução, mesmo que temporária, da dimensão da matriz A eliminando linhas e/ou colunas através de critérios heurísticos. Veremos a seguir algumas delas.

Cavalier & Schall [17] propõem uma heurística para a redução de restrições para problemas no seguinte formato:

$$\begin{aligned} \text{Max } & cx \\ \text{sa } & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Para transformar este problema na forma

padrão, basta acrescentar as variáveis de folga $s \geq 0$. A heurística proposta consiste em formar um conjunto Δ das restrições em que a variável de folga correspondente teve seu valor diminuído na iteração anterior. Assim, a dimensão das matrizes A_Δ e $A_\Delta D^2 A_\Delta^t$ é menor que a dimensão das matrizes originais. O conjunto Δ pode crescer quando alguma restrição não considerada for violada.

Uma heurística que diminui o número de colunas de A foi proposta por Lustig [59]. Ele propõe eliminar as variáveis que atingem um valor muito pequeno. Esta heurística não tem o mesmo efeito da anterior pois diminui o número de colunas de A e, portanto, a matriz AD^*A^t não tem sua dimensão alterada, embora seu cálculo fique menos trabalhoso. Note que uma heurística semelhante pode ser usada no algoritmo dual, neste caso, eliminando as variáveis de folga v que se aproximam de zero.

Gill e outros [34] propõem um esquema semelhante para restrições desprezando aquelas que na decomposição geram elementos muito pequenos na matriz diagonal ($E_i^i \leq \epsilon$).

Observação: Esta também é uma forma de tratar linhas redundantes na matriz AD^*A^t.

Uma forma de diminuir o número de colunas da matriz A no algoritmo dual é considerar apenas as variáveis de folga que diminuem, de forma similar a utilizada na implementação de Cavalier & Schall [17]. Esta heurística pode ocasionar situações interessantes quando o número de variáveis de folga consideradas em uma iteração for menor que o número de linhas da matriz. Neste caso, pode-se tratar o problema da mesma forma proposta por Gill e outros [34], ou escolher linhas da matriz que sejam mais interessantes de se trabalhar, por exemplo, as linhas que têm os maiores produtos $b_i y_i$ (onde y_i é a variável dual). A extensão desta heurística para o algoritmo dual afim canalizado é fácil, bastando considerar as variáveis duais z, além das variáveis de folga. Um levantamento do número de variáveis de folga cujo valor aumenta por iteração mostra que em alguns problemas, a redução do esforço computacional por iteração no algoritmo dual afim pode ser significativa.

3.8 Estrutura de Dados

A consideração da esparsidade exige estrutura de dados sofisticadas para o armazenamento das matrizes A, AD^*A^t e LEL^t. Uma vez que as operações realizadas pelas matrizes são diferentes, a forma de armazenamento também pode ser diferente.

As operações efetuadas pela matriz A são as seguintes:

- Multiplicação por um vetor;
- Multiplicação da transposta por um vetor;
- Reordenamento das linhas;
- Montagem de AD^*A^t.

As operações efetuadas por AD^*A^t são as seguintes:

- Cálculo dos elementos;
- Decomposição;
- Multiplicação por um vetor (método iterativo).

As operações efetuadas por LEL^t são as seguintes:

- Cálculo dos elementos (decomposição);
- Resolução de sistema triangular inferior;
- Resolução de sistema triangular superior;
- Atualização da decomposição.

Por medida de economia de memória, optou-se por armazenar as matrizes AD^*A^t e LEL^t nas mesmas posições. Da matriz AD^*A^t são armazenados somente os elementos da diagonal e da parte inferior. De LEL^t a matriz L é armazenada no triângulo inferior e a matriz E na diagonal de AD^*A^t.

Observações:

- é possível armazenar E na diagonal porque os elementos diagonais de L valem um.

- A matriz L^t fica armazenada implicitamente.

O armazenamento de E na diagonal facilita a execução de alguns procedimentos, pois garante que sempre haverá algum elemento não nulo armazenado em toda linha e coluna. A matriz L pode ter mais elementos que AD^*A^t (fill-ins), portanto é necessário diferenciar estes elementos na matriz. A presença deles não prejudica a resolução do sistema, pois, como eles precisam ser zerados antes da decomposição, isto pode ser feito no cálculo de AD^*A^t.

Está claro que o esquema de armazenamento deve permitir percorrer as matrizes por linha e por coluna. Optou-se pelo esquema de armazenamento proposto por Knuth [55] que tem estas facilidades. Porém, adotou-se uma inversão de apontadores na matriz L (figura 1-2) que permite resolver o sistema triangular superior mais facilmente. Outra vantagem desta estrutura é o uso de apontadores que agilizam o acesso à memória.

Esta estrutura pode ser desvantajosa em

aplicações que necessitam retirar ou acrescentar elementos da matriz. Este não é o caso do algoritmo estudado, pois a estrutura da matriz é a mesma durante toda a resolução do problema.

Além das matrizes, são utilizados dois vetores de apontadores com o objetivo de aumentar a eficiência da implementação: um vetor cujos elementos apontam para as cabeças de linhas da matriz A na ordem obtida pelo algoritmo de reordenamento; o outro vetor é utilizado para acessar rapidamente os elementos da matriz diagonal E.

Os elementos da matriz AD^*A^t têm ainda um campo a mais que contém o número de pares $A_i^k A_j^k$ diferentes de zero para sua formação. Note que quando o elemento for um fill-in de L, este campo conterá o valor zero, desta forma, se todos os pares $A_i^k A_j^k$ e o índice da coluna k forem armazenados em um vetor, a montagem dos elementos de AD^*A^t pode ser feita com a metade das multiplicações (item 3.3.5).

Existem muitas outras formas de armazenamento de matrizes esparsas que têm sido utilizadas com sucesso, algumas inclusive em implementações de métodos de pontos interiores [28,30,68,85].

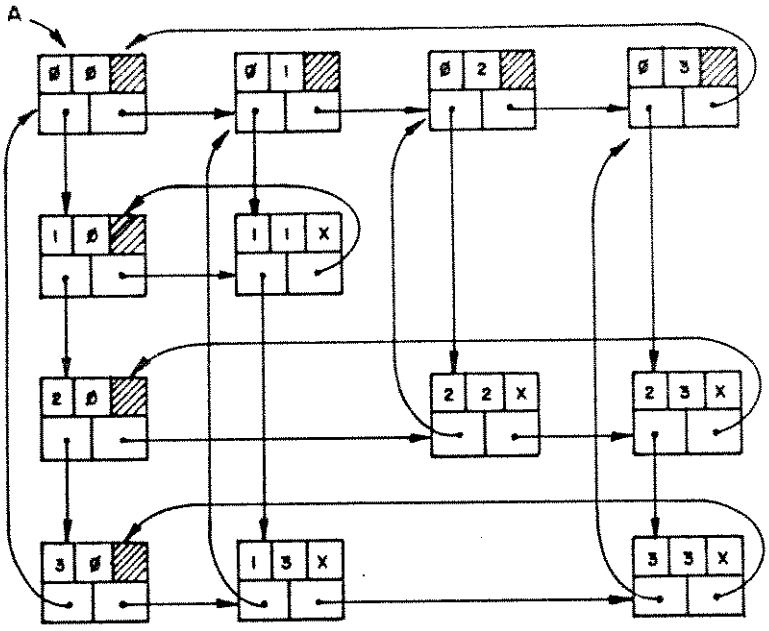


Fig. 1 – Estrutura de dados para a matriz A

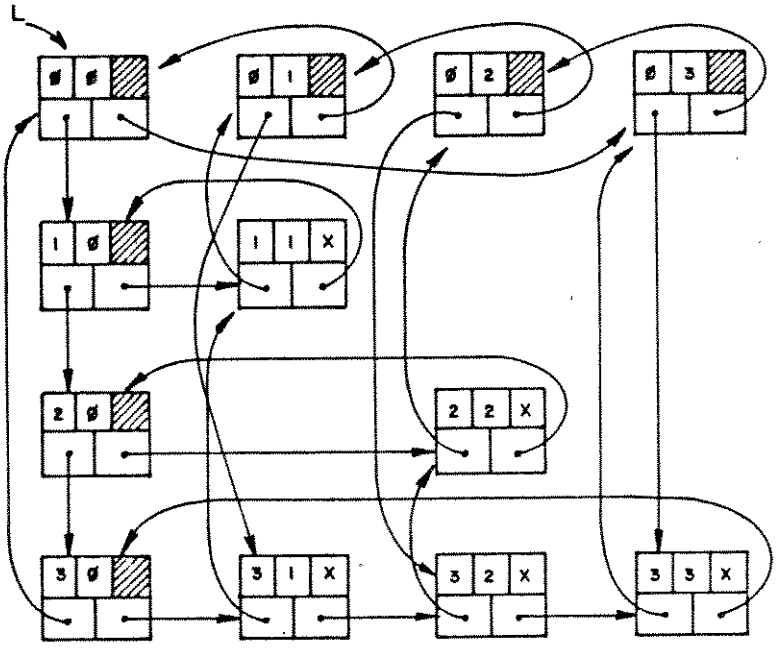


Fig. 2 – Estrutura de dados para a matriz L

$$A = \begin{bmatrix} x & & \\ & x & x \\ x & & x \end{bmatrix} \Longrightarrow L = \begin{bmatrix} x & & \\ & x & \\ x & x & x \end{bmatrix}$$

Capítulo 4

Resultados

4.1 Descrição da Implementação

Linguagem

Optou-se por utilizar a linguagem "C" devido a sua facilidade na utilização de apontadores, versatilidade, portabilidade, flexibilidade e velocidade de execução.

Algoritmo Básico

Foram implementadados os algoritmos primal afim e dual afim, com as seguintes características em comum: canalização, variação da função objetivo, condição de folgas complementares como critério de convergência e reordenamento da matriz A pelo método "minimum degree". A montagem de $AD^{*}A^{t}$ é feita aproveitando-se o produto $A_i^k A_j^k$ sempre que houver memória disponível. Quando não for o caso, o número de multiplicações na montagem da matriz fica duas vezes maior.

O sistema linear é resolvido na primeira iteração por decomposição LEL^t e nas seguintes pela opção que realize menos operações, uma nova decomposição ou atualização da anterior, conforme o item 3.4.4. Linhas redundantes e erros de arredondamento, na decomposição, são detectados quando $E^i_i < \epsilon_p$. O ponto inicial é calculado conforme as heurísticas do item 2.7.1.

A fase I utilizada no algoritmo primal é o método da variável livre descrito no item 2.7.4. Na fase II, é utilizado o método de diminuição do erro de arredondamento descrito no item 2.7.6.

Para o algoritmo dual, a fase I utilizada é o método do M grande com $M \rightarrow -\infty$ descrito no item 2.7.3. Porém, a fase I é evitada quando $c > 0$ ($y^o \leftarrow 0$), ou quando todas as variáveis primais forem canalizadas, conforme as heurísticas descritas no item 2.7.1. Se a variável de folga, ao final da fase I, tiver um valor muito pequeno (não existe ponto interior), utiliza-se um valor grande para M em conjunto com o vetor b original do problema e a "fase II" encontra um ótimo deste problema modificado.

Testes Realizados

Foram feitas algumas experiências comparando

a implementação com o simplex (MINOS 2.5) e com variações da implementação básica envolvendo: centragem, iterações continuadas, não tratamento implícito de canalização e eliminação de colunas.

Parâmetros

Os parâmetros de implementação utilizados são os seguintes (Não foi realizado um estudo sitemático para observar o comportamento do método em função dos parâmetros):

- Tamanho do passo na fase I: 0.99
- Tamanho do passo na fase II: 0.95
- Tamanho do passo na redução de infactibilidade: 0.95
- Precisão na variação da função objetivo: 10^{-8}
- Diferença relativa da variável de artificial: 10^{-5}
- Precisão nas folgas complementares: 10^{-5}
- Maior erro de arredondamento (primal afim): 10^{-5}
- Precisão na variação da função objetivo na redução: $\frac{1}{2}$
- Variação relativa de D_i^i para atualização: 0.1
- Valor da variável de folga no final da fase I: 10^{-5}
- Valor do menor pivô (E_i^i): 10^{-12}
- Precisão na busca unidimensional (centragem): 10^{-5}

4.2 Descrição dos Problemas

O estudo teórico do comportamento de um

algoritmo é fundamental para obter implementações eficientes. Por outro lado, é necessário nestes estudos fazer simplificações e, em particular para os algoritmos de nosso interesse (ponto interior e simplex) não se conhece nenhum resultado teórico de sua complexidade média. Em vista disto, comparações entre implementações diferentes para diversos problemas são úteis para a avaliação dos resultados teóricos obtidos e também para indicar caminhos para estudos posteriores.

A maioria dos problemas testados formam o primeiro conjunto de problemas utilizados em [01] (NETLIB problemas testes de programação linear [27]). Além destes, foram utilizados quatro problemas, que têm a característica de trabalhar com variáveis canalizadas. Os problemas são os seguintes:

1) CSF3

Problema de planejamento da operação de sistema hidrelétrico aplicado ao sistema do médio São Francisco [13,23].

2) ENER12G

Problema de coordenação da operação de um sistema multisetorial de suprimento de energia (problema de grafo generalizado) [21,22].

3)IEEE24

Problema de despacho ótimo de sistemas de potência aplicado ao sistema IEEE24 (grafo generalizado com restrições adicionais [15,16]).

4) CANTAR

Problema de operação ótima de um sistema de reservatórios aplicado ao sistema Cantareira (problema de fluxo de custo mínimo) [03].

4.3 Resultados Obtidos

Os resultados mostrados a seguir, para o conjunto de problemas NETLIB, foram obtidos em um VAX 11/785, sistema operacional VMS v4.7, utilizando os compiladores VAX C 2.3 para o método de ponto interior e VAX FORTRAN 4.6 para o MINOS, em ambos os casos, as opções "default" são utilizadas incluindo a otimização do código. Os parâmetros do MINOS são os usuais exceto: *rows*= 1500, *columns*= 6000, *elements*= 50000, *iterations*= 7500, *log frequency*= 200 e *solution*= NO. O outro conjunto de problemas foi testado em um microcomputador compatível ao IBM/PC, Cobra XPC v3.02 com coprocessador 8087, 8 MHz e compilador Turbo C 1.5.

PROBLEMA	LINHAS	COLUNAS	$\hat{A}$	$\hat{L}$	FILL-IN
Afiro	27	51	102	107	17
ADLittle	56	138	424	411	27
Scagr7	129	185	465	767	138
Sc205	205	317	665	1172	516
Share2b	96	162	777	1040	169
Share1b	117	253	1179	1383	382
Scorpion	388	466	1534	2559	458
Scagr25	471	671	1725	2981	588
ScTap1	300	660	1872	2620	934
BrandY	220	303	2202	3450	689
Scsd1	77	760	2388	1392	259
Israel	174	316	2443	11599	372
BandM	305	472	2494	4664	940
Scfxm1	330	600	2732	4715	1482
E226	223	472	2768	3687	864
Scrs8	490	1275	3288	6646	4448
Beaconfd	173	295	3408	2901	59
Scsd6	147	1350	4316	2545	446
Ship04s	402	1506	4400	3654	340
Scfxm2	660	1200	5469	9647	3161
Ship04l	402	2166	6380	4830	200
Ship08s	778	2467	7194	6238	732
ScTap2	1090	2500	7334	14805	8210
Scfxm3	990	1800	8206	14567	4828
Ship12s	1151	2869	8284	7576	1080
Scsd8	397	2750	8584	5879	1599
ScTap3	1480	3340	9734	18843	9977
CzProb	929	3562	10708	8391	390
25FV47	821	1876	10705	34789	22892
Ship08l	778	4363	12882	9714	424
Ship12l	1151	5533	16276	12328	504

Tab 1. - NETLIB

Os dados referem-se aos problemas na forma padrão. O elementos não nulos da matriz L incluem a diagonal. O número de "fill-ins" foi obtido utilizando o método de reordenamento "minimum degree" descrito no capítulo 3.

PROBLEMA	op(AD^*A^t)	op(Dec)	TEMPO DE REORDENAMENTO (seg)
Afiro	183	254	0,04
ADLittle	1241	1907	0,19
Scagr7	1071	2977	0,37
Sc205	1164	4071	0,64
Share2b	3706	6528	0,47
Share1b	4232	10581	0,68
Scorpion	4532	12338	1,66
Scagr25	4161	12175	1,94
ScTap1	4728	16919	1,70
BrandY	17611	51410	2,59
Scsd1	5336	15008	0,70
Israel	47379	528730	7,81
BandM	13964	47272	3,73
Scfxm1	14492	49160	3,30
E226	17668	43960	2,87
Scrs8	6651	97070	5,73
Beaconfd	36558	36155	2,53
Scsd6	9722	24853	1,35
Ship04s	8846	26252	2,61
Scfxm2	28883	102631	8,03
Ship04l	12806	36080	3,98
Ship08s	14487	43436	5,19
ScTap2	18868	287169	18,64
Scfxm3	43337	155736	13,59
Ship12s	49430	16605	7,79
Scsd8	19028	47803	3,29
ScTap3	25018	308714	25,98
CzProb	21638	56323	20,04
25FV47	1330830	50742	55,88
Ship08l	25863	73560	9,08
Ship12l	32593	90542	11,85

Tab. 2 - AD^*A^t

A coluna op(AD^*A^t) contém o número de multiplicações necessárias para calcular AD^*A^t. A coluna op(Dec) mostra o número de multiplicações e divisões para decompor a matriz AD^*A^t. O tempo de reordenamento inclui a identificação dos elementos não nulos de AD^*A^t a partir de A.

PROBLEMA	FASE I ITERAÇÕES	TOTAL DE ITERAÇÕES	TEMPO SEGS.	FUNÇÃO OBJETIVO
Afiro	7	9	0,20	-4,6475315e02
ADLittle	34	89	4,23	2,2549495e05
Scagr7	57	134	9,21	-2,3313897e06
Sc205	114	208	21,39	-5,2202057e01
Share2b	77	114	8,25	-4,1573227e02
Share1b	120	262	22,72	-7,6589298e04
Scorpion	91	148	18,45	1,8781247e03
Scagr25	250	680	144,35	-1,4753432e07
ScTap1	188	269	34,17	1,4122500e03
BrandY	234	388	50,93	1,5185098e03
Scsd1	110	242	21,59	8,6666666e00
Israel	102	331	47,41	-8,9664482e05
BandM	258	555	106,35	-1,5862801e02
Scfxm1	239	429	56,07	1,8416758e04
E226 *	127	599	87,18	-1,1638929e01
Scrs8	182	563	125,68	9,0429693e02
Beaconfd	15	48	6,49	3,3592485e04
Scsd6	183	605	96,63	5,0500000e01
Ship04s	97	240	43,11	1,7987146e06
Scfxm2	587	967	249,83	3,6660261e04
Ship04l	58	345	77,92	1,7933245e06
Ship08s	71	327	98,73	1,9200982e06
ScTap2	459	899	357,06	1,7248071e03
Scfxm3	874	1400	519,01	5,4901253e04
Ship12s	207	604	242,79	1,4892361e06
Scsd8	763	1604	622,85	9,0500000e02
ScTap3	479	1107	589,99	1,4240000e03
CzProb	809	1650	627,55	2,1851967e06
25FV47	1052	5445	4338,39	5,5018460e03
Ship08l	164	561	220,41	1,9090552e06
Ship12l	577	1367	720,90	1,4701879e06

Tab. 3 - MINOS 2.5

O tempo medido, da mesma forma que em [01], é o da sub-rotina DRIVER, que exclui a leitura e interpretação dos dados. Também como em [01], a solução do problema E226 obtida pelo MINOS está muito distante do ótimo. Os tempos dos métodos primal e dual afim incluem o reordenamento.

PROBLEMA	FASE I ITERAÇÕES	TOTAL DE ITERAÇÕES	TEMPO SEGS	FUNÇÃO OBJETIVO
Afiro	1	20	0,43	-4,6475316e02
ADLittle	1	24	2,20	2,2549495e05
Scagr7	3	26	3,64	-2,3313897e06
Sc205	5	30	6,27	-5,2202060e01
Share2b	4	28	6,65	-4,1573227e02
Share1b	8	38	12,95	-7,6589299e04
Scorpion	5	44	22,04	1,8781477e03
Scagr25	3	29	17,60	-1,4753432e07
ScTap1	6	36	22,89	1,4122499e03
BrandY *	6	233	308,74	1,3424202e03
Scsd1	0	300	143,88	8,6666660e00
Israel	9	39	423,43	-8,9664482e05
BandM	7	32	45,00	-1,5862801e02
Scfxm1	10	39	60,17	1,8416758e04
E226	12	51	66,37	-1,8751931e01
Scrs8 *	14	43	114,80	9,0429596e02
Beaconfd	9	31	42,42	3,3592485e04
Scsd6	0	20	18,94	5,0499998e01
Ship04s	5	30	31,99	1,7987146e06
Scfxm2	10	43	142,06	3,6659516e04
Ship04l	5	30	44,46	1,7933243e06
Ship08s	5	33	59,98	1,9200982e06
ScTap2	7	34	272,80	1,7248071e03
Scfxm3	10	42	214,46	5,4897951e04
Ship12s	5	40	84,89	1,4892361e06
Scsd8	0	21	39,66	9,0499999e02
ScTap3	7	36	341,32	1,4239999e03
CzProb	3	53	142,29	2,1825283e06
25FV47	11	59	1906,30	5,5018419e03
Ship08l	5	35	104,85	1,9090552e06
Ship12l	5	31	118,37	1,4701879e06

Tab. 4 - Método Dual Afim

O teste de convergência baseado nas folgas complementares não é realizado nos problemas BrandY e Scrs8 pois, nestes casos, ocorre "overflow". Por isso, o valor obtido para o problema BrandY está distante do ótimo. O número máximo de iterações em cada fase foi limitado em 300.

PROBLEMA	FASE I ITERAÇÕES	TOTAL DE ITERAÇÕES	TEMPO SEGS.	FUNÇÃO OBJETIVO
Afiro	3	20	0,51	-4,6475320e02
ADLittle	13	313	23,05	2,2604479e05
Scagr7	7	188	31,45	-2,3315204e06
Sc205	9	173	35,93	-5,2202086e01
Share2b	19	319	71,79	-4,1085889e02
Share1b	99	399	95,87	-4,3602829e04
Scorpion	15	315	153,23	1,8781202e03
Scagr25	19	319	230,59	-1,4721441e07
ScTap1	27	90	38,81	1,4122495e03
BrandY	18	318	230,37	1,5184862e03
Scsd1	1	301	215,80	9,3812292e00
Israel *	—	—	—	—
BandM *	300	—	199,72	—
Scfxm1	26	326	369,29	1,8417387e04
E226	29	329	418,10	-1,2867733e01
Scrs8 *	300	—	673,31	—
Beaconfd	16	39	53,78	3,3527123e04
Scsd6	1	301	311,85	7,8557848e09
Ship04s	15	315	352,34	1,7986837e06
Scfxm2	34	334	866,43	3,6660442e04
Ship04l	14	314	442,07	1,7933217e06
Ship08s	17	128	236,90	1,9201039e06
ScTap2	21	188	1309,76	1,7248072e03
Scfxm3	35	335	1333,41	5,4901213e04
Ship12s	18	197	414,96	1,4892882e06
Scsd8	1	301	611,61	1,3369040e09
ScTap3	19	183	1556,49	1,4239999e03
CzProb	30	284	828,44	2,1835727e06
25FV47	23	323	7917,17	5,5023964e03
Ship08l	15	315	846,40	1,9091883e06
Ship12l	17	129	496,10	1,4702400e06

Tab. 5 - Método Primal Afim

O método primal afim não encontrou uma solução factível para os problemas BandM e Scrs8 em 300 iterações e, para diversos problemas, atingiu o limite de iterações na fase II, Ocorreu "overflow" no problema Israel

PROBLEMA	FASE I ITERAÇÕES	TOTAL DE ITERAÇÕES	TEMPO SEGS.	FUNÇÃO OBJETIVO
Afiro	1	20	0,77	-4,6475315e02
ADLittle	1	25	3,61	2,2549495e05
Scagr7	3	28	5,96	-2,3313897e06
Sc205	5	30	10,08	-5,2202060e01
Share2b	4	27	9,28	-4,1573227e02
Share1b	8	41	18,58	-7,6589298e04
Scorpion	6	35	25,63	1,8781723e03
Scagr25	3	32	27,83	-1,4753433e07
ScTap1	7	37	33,34	1,4122499e03
BrandY	7	249	333,50	4,8010549e02
Scsd1	0	13	11,74	8,6666330e00
Israel	10	43	481,52	-8,9664482e05
BandM	7	38	64,23	-1,5862801e02
Scfxm1	11	45	84,57	1,8416758e04
E226	13	55	85,36	-1,8751931e01
Scrs8 *	15	—	—	—
Beaconfd	10	41	67,37	3,3592486e04
Scsd6	0	22	34,28	5,0499998e01
Ship04s	8	42	65,83	1,7987147e06
Scfxm2	11	51	199,90	3,6660260e04
Ship04l	7	40	86,47	1,7933244e06
Ship08s	7	40	102,58	1,9200982e06
ScTap2	8	39	359,53	1,7248071e03
Scfxm3	11	51	307,15	5,4890125e04
Ship12s	7	43	134,86	1,4892361e06
Scsd8	0	21	68,70	9,0499999e02
ScTap3	8	41	444,76	1,4239999e03
CzProb	2	63	201,93	2,1825283e06
25FV47	13	69	2291,56	5,5018459e03
Ship08l	7	42	174,36	1,9090552e06
Ship12l	9	42	232,19	1,4701879e06

Tab. 6 - Método Dual Afim com Centragem

A centragem é utilizada nas 20 primeiras iterações das fases I e II. O critério de convergência baseado nas folgas complementares não é utilizado. Ocorreu "overflow" no problema Scrs8.

PROBLEMA	FASE I ITERAÇÕES	TOTAL DE ITERAÇÕES	TEMPO SEGS.	FUNÇÃO OBJETIVO
Afiro	1	17	0,73	-4,6475315e02
ADLittle	1	21	3,50	2,2549495e05
Scagr7	3	22	5,22	-2,3313897e06
Sc205	5	26	9,27	-5,2202059e01
Share2b	4	24	8,48	-4,1573227e02
Share1b	8	38	17,25	-7,6589298e04
Scorpion	5	28	21,70	1,8781611e03
Scagr25	3	23	23,80	-1,4753433e07
ScTap1	6	31	28,95	1,4122499e03
BrandY	6	72	98,63	-1,4206230e03
Scsd1	0	12	11,60	8,6666500e00
Israel	9	38	425,80	-8,9664482e05
BandM	7	32	55,25	-1,5862801e02
Scfxm1	10	40	74,66	1,8416758e04
E226	12	45	70,66	-1,8751930e01
Scrs8	14	44	137,88	9,0429681e02
Beaconfd	9	33	55,30	3,3592485e04
Scsd6	0	18	31,76	5,0499999e01
Ship04s	5	34	56,12	1,7987146e06
Scfxm2	10	46	180,26	3,6660261e04
Ship04l	5	36	80,57	1,7933245e06
Ship08s	5	38	100,43	1,9200982e06
ScTap2	7	34	312,96	1,7248071e03
Scfxm3	10	43	261,16	5,4901253e04
Ship12s	5	37	117,88	1,4892361e06
Scsd8	0	18	65,18	9,0499999e02
ScTap3	7	36	396,30	1,4239999e03
CzProb	3	93	300,04	2,1825175e06
25FV47	11	60	2022,85	5,5018411e03
Ship08l	5	39	168,29	1,9090552e06
Ship12l	5	37	209,37	1,4701879e06

Tab. 7 - Iterações Continuadas com Centragem

Após uma iteração da fase II, é feita uma centragem e realizado novamente um passo na mesma direção da iteração. O processo é repetido nas 20 primeiras iterações. O critério de convergência baseado nas folgas complementares não é utilizado.

PROBLEMA	FASE I ITERAÇÕES	TOTAL DE ITERAÇÕES	TEMPO SEGS.	FUNÇÃO OBJETIVO
Afiro	1	21	0,46	-4,6475315e02
ADLittle	1	35	3,04	1,9416422e05
Scagr7	3	28	3,97	-2,3313897e06
Sc205	5	32	6,86	-5,2202059e01
Share2b	4	33	7,79	-4,1573227e02
Share1b	8	42	14,25	-7,6589298e04
Scorpion	5	26	14,04	1,8781434e03
Scagr25	3	40	23,87	-1,4753432e07
ScTap1	6	35	22,60	1,4122500e03
BrandY	6	300	389,06	1,2245177e03
Scsd1	0	33	16,55	8,6666510e00
Israel	9	48	429,27	-8,9664483e05
BandM	7	34	47,67	-1,5862801e02
Scfxm1	10	63	93,40	1,8347032e04
E226	12	51	67,17	-1,8751931e01
Scrs8	14	44	118,04	9,0429649e02
Beaconfd	9	36	47,98	3,3592485e04
Scsd6	0	20	19,82	5,0499999e01
Ship04s	5	31	33,99	1,7987129e06
Scfxm2	10	38	111,53	3,5963547e04
Ship04l	5	34	50,93	1,7933231e06
Ship08s	5	55	94,52	1,9200981e06
ScTap2	7	39	303,26	1,7248071e03
Scfxm3	10	76	361,66	5,4897055e04
Ship12s	5	40	86,44	1,4892361e06
Scsd8	0	22	43,15	9,0499999e02
ScTap3	10	39	368,28	1,4239999e03
CzProb	3	60	161,18	2,1825284e06
25FV47	11	55	1796,44	5,5018459e03
Ship08l	5	42	124,64	1,9090552e06
Ship12l	5	39	147,43	1,4701879e06

Tab. 8 — Iterações Continuadas

Após a iteração, anula-se as componentes da direção das variáveis duais dy que formam a direção da variável de bloqueio e realiza-se outro passo. O processo é repetido nas 20 primeiras iterações. O critério de convergência baseado nas folgas complementares não é usado.

PROBLEMA	FASE I			FASE II		
	TOTAL	ELIM.	MÉDIA	TOTAL	ELIM.	MÉDIA
Afiro	1	0	0	20	1	8
ADLittle	1	0	0	25	6	133
Scagr7	4	1	425	25	2	41
Sc205	5	2	52	27	2	27
Share2b	6	1	1778	25	1	182
Share1b	9	1	1697	34	1	705
Scorpion	6	2	685	18	1	295
Scagr25	4	1	1911	26	2	56
ScTap1	9	2	1938	31	1	1088
BrandY	8	1	6719	132	1	7487
Scsd1	0	0	0	12	3	2917
Israel	9	1	15641	31	1	4672
BandM *	3	1	5245	–	–	–
Scfxm1	11	1	5399	30	1	7327
E226	13	1	7988	39	1	7913
Scrs8	15	1	2244	31	1	5302
Beaconfd *	3	1	18030	–	–	–
Scsd6	0	0	0	20	4	5387
Ship04s	6	1	5480	24	2	1189
Scfxm2	11	1	10898	33	1	14342
Ship04l	6	1	8078	26	2	1723
Ship08s	6	1	9296	34	3	4432
ScTap2	8	1	5951	31	1	4370
Scfxm3	11	1	16432	33	1	20672
Ship12s	6	1	10329	36	1	3749
Scsd8	0	0	0	20	4	10386
ScTap3	8	1	7360	35	1	5670
CzProb	4	1	10589	55	5	8249
25FV47	13	1	24633	55	1	22462
Ship08l	6	1	15869	29	2	6594
Ship12l	6	1	19932	25	1	6405

Tab. 9 - Eliminação de Variáveis

As colunas cuja variável de folga tiveram o valor aumentado não são utilizadas no cálculo de $AD^{*}A^{t}$ da próxima iteração. O processo é repetido nas 15 primeiras iterações das fases I e II, ou até uma variação da função

objetivo menor que 0,1. Os tempos não são comparados porque a montagem de $AD^{*}A^{t}$ é feita por linha (de A) e não por coluna, que é a situação onde deverá haver ganho no tempo. Portanto, a comparação utilizada será a economia no número de operações. A coluna ELIM mostra o número de iterações em que houve eliminação de variáveis. A coluna MÉDIA contém o número médio de multiplicações economizadas nas iterações em que houve eliminação de variáveis. O método não encontrou solução factível para os problemas BandM e Beaconfd.

PROBLEMA	LINHAS	COLUNAS	$\hat{A}$	$\hat{L}$	FILL-IN
CSF3	168	294	566	779	205
Ener12G	460	684	1337	1792	403
IEEE24	91	114	257	364	30
Cantar	178	249	498	647	120

PROBLEMA	op($AD^{*}A^{t}$)	op(Dec)	TEMPO DE REORDENAMENTO (seg)
CSF3	1045	2402	2,25
Ener12G	2266	4953	7,47
IEEE24	560	918	1,04
Cantar	868	1468	2,03

PROBLEMA	FASE I ITERAÇÕES	TOTAL DE ITERAÇÕES	TEMPO SEGS.	FUNÇÃO OBJETIVO
CSF3	1	24	29,93	-4,3000000e02
Ener12G	2	25	66,62	-0,0000100e00
IEEE24	1	20	11,81	-1,9620000e03
Cantar	1	24	23,62	-2,0930501e02

Tab. 10 - Canalização na Matriz

A tabela 10 mostra os dados e resultados obtidos com o segundo subconjunto de problemas considerando as restrições de canalização como parte da matriz A. A tabela 11 mostra os dados considerando implicitamente a canalização.

PROBLEMA	LINHAS	COLUNAS	$\hat{A}$	$\hat{L}$	FILL-IN
CSF3	63	189	356	414	205
Ener12G	155	379	727	906	403
IEEE24	34	57	143	164	30
Cantar	57	128	256	284	120

PROBLEMA	op($AD^{*}A^{t}$)	op(Dec)	TEMPO DE REORDENAMENTO (seg)
CSF3	575	1675	1,21
Ener12G	1075	3515	3,24
IEEE24	303	472	0,44
Cantar	384	863	0,82

PROBLEMA	FASE I ITERAÇÕES	TOTAL DE ITERAÇÕES	TEMPO SEGS	FUNÇÃO OBJETIVO
CSF3	1	37	29,22	-4,3000002e02
Ener12G	2	21	38,83	-0,0000120e00
IEEE24	0	22	7,85	-1,9620001e03
Cantar	1	23	14,23	-2,0930501e02

Tab. 11 - Canalização fora da Matriz

Capítulo 5

Comentários e Conclusões

Os resultados obtidos no capítulo anterior mostram que o método de ponto interior dual afim concorre e até supera o método simplex em vários problemas. Isto vale principalmente para problemas esparsos de maior porte. Dois fatores são particularmente importantes para a boa performance do método:

- O pequeno número de iterações para a convergência e seu crescimento lento (o número médio de iterações obtido está muito abaixo do valor calculado para o pior caso);
- A facilidade de encontrar uma solução factível inicial (em alguns problemas não é necessária a fase I). Muitas vezes o processo de inicialização é o que toma a maior parte do tempo utilizado pelo simplex.

Por outro lado, em algumas classes de problemas o método simplex ainda é o mais indicado. Por exemplo, problemas com a presença de colunas densas como é o caso do problema ISRAEL. A utilização de métodos iterativos para a resolução do sistema linear, melhora em muito a eficiência do método de ponto interior com colunas densas [01], porém isto não parece ser suficiente para superar o método simplex. A implementação do gradiente conjugado

pré-condicionado utilizada obteve resultados inferiores em relação à decomposição mesmo no problema ISRAEL, porque, na maioria das iterações, o método do gradiente conjugado demorou a convergir. O pré-condicionamento usado em [01] obteve melhores resultados.

O método dual afim mostrou-se mais robusto que o primal afim, obtendo melhor performance na grande maioria dos problemas (a exceção é o problema BrandY). O método primal afim mostrou também maior dificuldade para encontrar uma solução factível inicial, assemelhando-se neste ponto ao simplex. Outra desvantagem é a dificuldade de manter um ponto factível a cada iteração ($Ax^k = b$) com uma precisão aceitável. A utilização de um passo de "fase I" na fase II, aproveitando a mesma decomposição resolve, em alguns casos este problema ao preço de aumentar o número de iterações e tornar a iteração do método primal mais cara em relação ao método dual. Ainda assim, os resultados obtidos pelo método primal afim não foram satisfatórios para uma grande maioria de problemas, sendo sua performance inferior ao MINOS.

Os problemas de "overflow" podem ser suprimidos utilizando uma opção para representação de números reais mais precisa. Isto pode ser evitado no dual afim, pois o problema ocorreu no cálculo da norma para o

teste da condição de folgas complementares, assim, basta dividir os elementos do vetor pelo elemento de maior valor, ou ainda utilizar uma outra norma, por exemplo, norma 1 ou norma infinita.

Os resultados obtidos por Gill e outros [34], com o método das barreiras na forma primal para um subconjunto dos problemas testados, são muito melhores que os obtidos pelo método primal afim. Por outro lado, o critério de convergência utilizado em [34] é menos rígido. Uma das causas desta boa performance parece ser um melhor tratamento de linhas redundantes e pivôs de pequeno valor na decomposição, através da eliminação das linhas correspondentes.

Os problemas que não têm ponto interior na forma dual (BrandY, Scfxm1, E226, Scrs8, Beaconfd, Scfxm2, Scfxm3 e 25FV47) tiveram seus tempos aumentados devido a uma "fase II" para a determinação da solução ótima. Isto não ocorre nas implementações que utilizam o método de inicialização sugerido em [01], que encontra, nestes casos, a solução ótima na fase I. Por outro lado, o tempo das iterações na fase I nesta implementação é menor por trabalhar com as dimensões originais do problema, portanto, há ganho em tempo nos problemas que têm ponto interior.

O método dual afim obteve resultados ruins para os problemas Scsd1 e BrandY. A realização de um passo com 90% do valor máximo consegue resultados muito melhores para o problema Scsd1. O mau comportamento do método para o problema Brandy também está ligado a existência de linhas redundantes.

Das sofisticações testadas, a atualização da decomposição é indicada sempre que for menos cara que uma nova decomposição (o que depende das condições de cada iteração), pois a atualização diminui o tempo da iteração sem comprometer a convergência do método dual afim. A centragem e as iterações continuadas não parecem interessantes de serem usadas sistematicamente, pois, em muitos casos, o número de iterações pode aumentar. Elas parecem mais indicadas em problemas que realizam passos muito pequenos, ainda longe da solução ótima. Destes métodos, o que parece mais promissor, é a centragem com mais um passo na iteração, pois, ele obteve a maior redução no número total de iterações. Uma melhoria na eficiência do procedimento de centragem talvez torne esta variação mais rápida que o algoritmo básico.

A eliminação de restrições por sua vez, será mais útil em situações específicas onde se conhece melhor a estrutura do problema tratado. Nota-se pelos resultados que

não é recomendável utilizar a eliminação de variáveis na fase I, pois como o objetivo desta fase é diminuir o valor da variável artificial, as variáveis de folga têm a tendência de crescimento e a sua eliminação em uma iteração tem como consequência o aumento do total de iterações na fase I, ou até mesmo a conclusão equivocada que o problema é dual infactível.

A redução de infactibilidade melhora a performance do método primal afim para alguns problemas, reduzindo o número de iterações na fase I, mas em muitos casos compromete ainda mais a convergência do método.

A utilização de variáveis canalizadas implicitamente resultou em um ganho efetivo no número de operações por iteração, confirmado pelos tempos obtidos nos casos teste, onde, em alguns casos, a versão canalizada obteve tempos inferiores mesmo quando convergiu em um número maior de iterações. A diferença do número de iterações nestes casos (canalização implícita ou explícita) se deve ao ponto inicial ser obtido de forma distinta.

Uma observação interessante é que as restrições de canalização colocadas na matriz A não geram novos "fill-ins" em $AD^{*}A^{t}$, se for utilizado o reordenamento "minimum degree". Isto porque no grafo associado a matriz

AD^*A^t, o nó que representa uma restrição de canalização forma um subgrafo completo com seus nós adjacentes. Neste subgrafo, ou o nó dessa restrição é escolhido primeiro (e não gera "fill-in" pois pertence a um grafo completo) ou algum outro nó, que pertence a este grafo completo, é escolhido primeiro. Neste caso, o grau deste nó é igual ao do nó que representa a restrição de canalização, ou seja, eles estão ligados aos mesmos nós e a escolha de qualquer deles não gera "fill-ins". Finalmente, a escolha de um nó que não pertence ao subgrafo não gera "fill-ins" adicionais com a linha de restrição de canalização.

O tempo de solução para problemas de custo de fluxo mínimo, grafo generalizado e grafo com restrições adicionais são muito superiores aos obtidos por implementações específicas do simplex [16,22]. Uma especialização do algoritmo de ponto interior e principalmente a resolução de problemas de maior porte podem reduzir esta diferença e talvez, em alguns casos, revertê-la.

Futuros Desenvolvimentos

Os próximos passos a serem desenvolvidos a partir deste trabalho são:

- Eliminação de variáveis fixas;
- Aumento da esparsidade da matriz A;
- Implementação de variantes do método baseadas em funções não-lineares;
- Implementação de outras heurísticas de reordenamento;
- Desenvolvimento e implementação de métodos para estruturas específicas;
- Estudo de um método primal-dual;
- Estudo da aplicação do método a funções não lineares;
- Integração da implementação com uma rotina simplex;
- Eliminação de linhas redundantes;
- Incorporação de aspectos de análise de sensibilidade ao método;
- Algoritmo parametrizado;
- Integração com um algoritmo tipo "Branch and Bound".

Apêndice A

Alguns Resultados de Álgebra Matricial

O objetivo deste apêndice é apresentar a demonstração de algumas identidades matriciais utilizadas no texto. Uma visão mais completa do assunto pode ser encontrada em vários livros, por exemplo, [38,66].

As demonstrações a seguir assumem que as dimensões das matrizes envolvidas são compatíveis, e que as matrizes que necessitam inversa são quadradas e não singulares.

Identidade (I1)

$$\mathcal{A}\mathcal{B}(\mathcal{C}^{-1} + \mathcal{B}^{t}\mathcal{A}\mathcal{B})^{-1} = (\mathcal{A}^{-1} + \mathcal{B}\mathcal{C}\mathcal{B}^{t})^{-1}\mathcal{B}\mathcal{C}$$

pré e pós-multiplicando pela matriz identidade, vem:

$$(\mathcal{A}^{-1} + \mathcal{B}\mathcal{C}\mathcal{B}^{t})^{-1}(\mathcal{A}^{-1} + \mathcal{B}\mathcal{C}\mathcal{B}^{t})\cdot\mathcal{A}\mathcal{B}(\mathcal{C}^{-1} + \mathcal{B}^{t}\mathcal{A}\mathcal{B})^{-1} =$$
$$(\mathcal{A}^{-1} + \mathcal{B}\mathcal{C}\mathcal{B}^{t})^{-1}\mathcal{B}\mathcal{C}\cdot(\mathcal{C}^{-1} + \mathcal{B}^{t}\mathcal{A}\mathcal{B})(\mathcal{C}^{-1} + \mathcal{B}^{t}\mathcal{A}\mathcal{B})^{-1}$$

$$(\mathcal{A}^{-1} + \mathcal{B}\mathcal{C}\mathcal{B}^{t})^{-1}(\mathcal{B} + \mathcal{B}\mathcal{C}\mathcal{B}^{t}\mathcal{A}\mathcal{B})(\mathcal{C}^{-1} + \mathcal{B}^{t}\mathcal{A}\mathcal{B})^{-1} =$$
$$(\mathcal{A}^{-1} + \mathcal{B}\mathcal{C}\mathcal{B}^{t})^{-1}(\mathcal{B} + \mathcal{B}\mathcal{C}\mathcal{B}^{t}\mathcal{A}\mathcal{B})(\mathcal{C}^{-1} + \mathcal{B}^{t}\mathcal{A}\mathcal{B})^{-1}$$

$$(A^{-1} + BCB^t)^{-1}BC(C^{-1} + B^tAB)(C^{-1} + B^tAB)^{-1} =$$
$$(A^{-1} + BCB^t)^{-1}(A^{-1} + BCB^t)AB(C^{-1} + B^tAB)^{-1}$$

$$(A^{-1} + BCB^t)^{-1}BC = AB(C^{-1} + B^tAB)^{-1} \quad \square$$

Identidade (I2)

$$(C^{-1} + B^tAB)^{-1} = C - CB^t(A^{-1} + BCB^t)^{-1}BC$$

pré-multiplicando por $(C^{-1} + B^tAB)$, vem:

$$(C^{-1} + B^tAB)(C^{-1} + B^tAB)^{-1} = (C^{-1} + B^tAB)(C - CB^t(A^{-1} + BCB^t)^{-1}BC)$$
$$I = I - B^t(A^{-1} + BCB^t)^{-1}BC + B^tABC - B^tABCB^t(A^{-1} + BCB^t)^{-1}BC$$
$$I = I - B^t((A^{-1} + BCB^t)^{-1} - A + ABCB^t(A^{-1} + BCB^t)^{-1})BC$$
$$I = I - B^t((I + ABCB^t)(A^{-1} + BCB^t)^{-1} - A)BC$$
$$I = I - B^t(A(A^{-1} + BCB^t)(A^{-1} + BCB^t)^{-1} - A)BC$$
$$I = I - B^t(A - A)BC = I \quad \square$$

Atualização da Inversa

(Fórmula de Shermann-Morrisson-Woodbury)

$$(A + CR)^{-1} = A^{-1} - A^{-1}CK^{-1}RA^{-1}$$

Onde $K = I + RA^{-1}C$

pré-multiplicando por $(A + CR)$, vem:

$$(A + CR)(A + CR)^{-1} = (A + CR)(A^{-1} - A^{-1}CK^{-1}RA^{-1})$$
$$I = I - CK^{-1}RA^{-1} + CRA^{-1} - CRA^{-1}CK^{-1}RA^{-1}$$

$$= I - \mathcal{C}(-(I + \mathcal{R}\mathcal{A}^{-1}\mathcal{C})\mathcal{K}^{-1} + I)\mathcal{R}\mathcal{A}^{-1}$$

$$= I - \mathcal{C}(-\mathcal{C}\mathcal{K}^{-1} + I)\mathcal{R}\mathcal{A}^{-1} = I_{\square}$$

A atualização de "Rank 1" (fórmula de Shermann-Morrison) é um caso especial da atualização da inversa onde c e r são vetores.

$$(\mathcal{A} + cr)^{-1} = A^{-1} - \frac{A^{-1}crA^{-1}}{1 + rA^{-1}c}$$

Resolução do Sistema Modificado

Seja $\mathcal{A}x = b$, então:

$$(\mathcal{A} + \mathcal{C}\mathcal{R})x' = b$$

$$x' = (\mathcal{A} + \mathcal{C}\mathcal{R})^{-1}b$$

$$x' = (\mathcal{A}^{-1} + \mathcal{A}^{-1}\mathcal{C}\mathcal{K}^{-1}\mathcal{R}\mathcal{A}^{-1})b$$

$$x' = x - \mathcal{A}^{-1}\mathcal{C}\mathcal{K}^{-1}\mathcal{R}x$$

Bibliografia

[01] Adler I. & Karmarkar N. & Resende M. G. C. & Veiga G. "An Implementation of Karmarkar's Algorithm for Linear Programming" University of California, Berkeley 1986.

[02] Aho A. W. & Hopcroft J. E. & Ullman J. D. *The Design and Analysis of Computer Algorithms*, Addison-Wesley 1975.

[03] Andrade M. G. & Correia P. B. "Operação ótima de um Sistema de Reservatórios com Algoritmo em Rede" Anais do 7º Congresso Brasileiro de Automática (1988) 2, 872-879.

[04] Anstreicher K. M. "A Monotonic Projective Algorithm for Fractional Linear Programming" Algorithmica (1986) 1, 483-498.

[05] Arantes R. D. & Tortorelli M. M. F. "Implementação e Estudo Comparativo de Algoritmos de Pontos Interiores para Programação Linear" Manuscrito COPPE-PUC/RJ 1988.

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

[07] Barnes E. R. & Chopra S. & Jensen D. L. "A Polynomial Time Version of the Affine Scaling Algorithm" IBM Thomas J. Watson Research Center 1988.

[08] Bazaraa M. S. & Jarvis J. J. *Linear Programming and Networks Flows*, John Wiley & Sons 1977.

[09] Benichou M. & Gauthier J. M. & Hentges G. & Ribiere G. "The Efficient Solution of Large-Scale Linear Programming Problems -- Some Algorithmic Techniques and Computational Results" Mathematical Programming (1977) 13, 280-322.

[10] Bennett J. M. "Triangular Factors of Modified Matrices" Numerische Mathematik (1965) 7, 217-221.

[11] Blair C. E. "The Iterative Step in the Linear Programming Algorithm of N. Karmarkar" Algorithmica (1986) 1, 537-539.

[12] Bland R. G. & Goldfarb D. & Todd M. J. "The Ellipsoid Method: A Survey" Operations Research (1981) 29, 1039-1091.

[13] Carneiro M. S. "Modelo de Otimização para a Operação Hidroelétrica da Cascata do S. Francisco" - Tese de Mestrado, FEE UNICAMP 1984.

[14] Carvalho M. "On the Minimization of Work Needed to Factor a Symmetric Positive Definite Matrix" University of California, Berkeley 1987.

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

[16] Carvalho M. F. H. & Soares S. & Ohishi T. "Optimal Active Power Dispatch by Network Flow Approach" IEEE Transactions on Power Systems (1988) 3, 1640-1647.

[17] Cavalier T. M. & Schall K. C. "Implementing an Affine

Scaling Algorithm for Linear Programming" Comput. Oprs. Res. (1987) 14, 341-347.

[18] Cavalier T. M. & Soyster A. L. "Some Computational Experience and a Modification of the Karmarkar Algorithm" 12th Int. Symp. Mathematical Programming 1985.

[19] Chandru V. & Kochar B. S. "A Class of Algorithms for Linear Programming" Purdue University 1986.

[20] Chandru V. & Kochar B. S. "Exploiting Special Structures Using a Variant of Karmarkar's Algorithm" Purdue University 1986.

[21] Correia P. B. "Um Modelo Multisetorial para Otimização do Suprimento de Energia: Eletricidade, Gás Natural e Cogeração com Biomassa" - Tese de Doutorado, FEE UNICAMP 1989.

[22] Correia P. B. & Lyra C. "Um algoritmo em Rede de Fluxo não-conservativa para Otimização do Suprimento de Energia" Anais do III Congresso Latino-Americano de Automática (1988) 1, 210-215, Viña del Mar, Chile.

[23] Correia P. B. & Lyra C. & Oliveira A. R. L. "Uma Implementação Computacional de Algoritmo Polinomial de Programação Linear: Aplicação ao Planejamento da Operação de Sistemas Hidrotérmicos" Anais do 7º Congresso Brasileiro de Automática (1988) 2, 924-929. ITA, S. J. Campos SP.

[24] Dantzig G. B. *Linear Programming and Extensions*

Princeton University Press 1963.

[25] Dennis J. E. & Morshedi A. M. & Turner K. "A Variable-Metric Variant of the Karmarkar Algorithm for Linear Programming" Mathematical Programming (1987) 39, 1-20.

[26] Dikin I. I. "Iterative Solution of Problems of Linear and Quadratic Programming" Soviet Mathematics Doklady (1967) 8, 674-675.

[27] Dongarra J. J. & Grosse E. "Distribuition of Mathematical Software via Eletronic Mail" Communications of the ACM (1987) 403-414.

[28] Duff I. S. & Erisman A. M. & Reid J. K. *Direct Methods for Sparse Matrices*, Clarendon Press 1986.

[29] Eisenstat S. C. "Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods" SIAM Journal of Scientific and Statical Computing (1981) 2, 1-4.

[30] Eisenstat S. C. & Schultz M. H. & Sherman A. H. "Algorithms and Data Structures for Sparse Symmetric Gaussian Elimination" SIAM Journal Science & Stat. Comp. (1981) 2, 225-237.

[31] Fletcher R. & Powell M. J. D. "On the Modification of LDLt Fatorizations" Mathematics of Computation (1974) 28, 1067-1087.

[32] Gay D. M. "A Variant of Karmarkar's Linear Programming Algorithm for Problems in Standard Form" Mathematical Programming (1987) 37, 81-90.

[33] Gill P. E. & Murray W. & Wright M. H. *Pratical Opmization*, Academic Press 1981.

[34] Gill P. E. & Murray W. & Saunders M. A. & Tomlin J. A. & Wright M. H. "On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method" Mathematical Programming (1986) 36, 183-209.

[35] Gill P. E. & Murray W. & Saunders M. A. & Wright M. H. "A Note on Nonlinear Approaches to Linear Programming" Stanford University 1986.

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

[37] Goldfarb D. & Todd M. J. "Linear Programming" Cornell University 1988.

[38] Golub G. H. & Van Loan C. F. *Matrix Computations*, The Johns Hopkins University Press 1983.

[39] Gonzaga C. C. "An Algorithm for Solving Linear Programming Problems in $O(n^aL)$ Operations" University of California, Berkeley 1987.

[40] Gonzaga C. C. "Search Directions for Interior Linear Programming Methods" University of California, Berkeley 1987.

[41] Gonzaga C. C. "Algoritmos de Pontos Interiores para Programação Linear" Minicurso 4, IV Congresso

Latino-Ibero-Americano de Pesquisa Operacional e Engenharia de Sistemas, XXI SOBRAPO, RJ. 1988.

[42] Gonzaga C. C. "Polynomial Affine Algorithms for Linear Programming" COPPE 1988.

[43] Gonzaga C. C. "Interior Point Algorithms for Linear Programming Problems With Inequality Constraints" COPPE 1987.

[44] Gonzaga C. C. "Um Algoritmo Afim Polinomial para Programação Linear" Pesquisa Operacional (1988) 8, 41-51.

[45] Gustavson F. G. & Liniger W. & Willoughby R. "Symbolic Generation of an Optimal Crout Algorithm for Sparse Systems of Linear Equations" Journal of the Association for Computing Machinery (1970) 17, 87-109.

[46] Heath M. T. "Numerical Methods for Large Sparse Linear Least Squares Problems" SIAM Journal of Scientifc and Statical Computing (1984) 5, 497-503.

[47] Hillien F. F. & Lieberman G. J. *Introdution to Operations Research* 3ª Edição, Holden-Day, inc. 1980.

[48] Hooker J. N. "Karmarkar's Linear Programming Algorithm" Interfaces (1986) 16, 75-90.

[49] Housos E. C. & Irisarri G. D. "A Sparse Variable Metric Optimization Method Applied to the Solution of Power System Problems" IEEE Transactions on Power Apparatus and Systems (1982) 101, 195-202.

[50] Iri M. & Imai H. "A Multiplicative Barrier Function

Method for Linear Programming" Algorithmica (1986) 1, 455-482.

[51] Irisarri G. D. & Sasson A. M. "An Automatic Contigency Selection Method for On-Line Security Analysis" IEEE Transactions on Power Apparatus Systems (1981) 100, 1838-1844.

[52] Karmarkar N. "A New Polynomial-Time Algorithm for Linear Programming" Proc. of the 16th Annual ACM Symp. on Theory of Computing (1984) 302-311.

[53] Karmarkar N. "A New Polynomial-Time Algorithm for Linear Programming" Combinatorica (1984) 4, 373-395.

[54] Khachiyan L. G. "A Polynomial Algorithm in Linear Programming" Soviet Mathematics Doklady (1979) 20, 191-194.

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

[56] Kojima M. "Determining Basic Variables of Optimal Solutions in Karmarkar's New LP Algorithm" Algorithmica (1986) 1, 499-515.

[57] Kojima M. & Mizuno S. & Yoshise A. "A Primal-Dual Interior Point Algorithm for Linear Programming" Tokyo Institute of Technology 1987.

[58] Luenberger D. G. *Linear and Nonlinear Programming*, Addison-Wesley Publishing Company 1984.

[59] Lustig I. J. "A Practical Approach to Karmarkar's

Algorithm" Stanford University 1985.

[60] Martinez J. M. "Generating Inexact-Newton Methods Using Least Change Secant Update Procedures" Workshop on Mathematical Programming 1988.

[61] Megiddo N. "Pathways to the Optimal Set in Linear Programming" IBM San Jose 1986.

[62] Monma C. L. & Morton A. J. "Computational Experience With a Dual Affine Variant of Karmarkar's Method For Linear Programming" Operations Research Letters (1987) 6, 261-267.

[63] Murty K. G. "The Gravitational Method for Linear Programming" Opsearch (1986) 25, 206-214.

[64] Nazareth J. L. "Homotopy Techniques in Linear Programming" Algorithmica (1986) 1, 529-535.

[65] Nickels W. & Rodder W. Xu L. & Zimmermann H. J. "Intelligent Gradient Search in Linear Programming" European Journal of Operational Research (1985) 22, 293-303.

[66] Noble B. & Daniel J. W. *Álgebra Linear Aplicada*, Prentice Hall do Brasil 1986.

[67] Padberg M. "A Different Convergence Proof of the Projective Method for Linear Programming" Operations Research Letters (1986) 4, 253-257.

[68] Paige C. C. & Saunders M. A. "LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares" ACM Transactions on Mathematical Software (1982) 8, 43-71.

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

[70] Renegar J. "A Polynomial-Time Algorithm, Based on Newton's Method, for Linear Programming" Mathematical Programming (1988) 40, 59-93.

[71] Rinaldi G. "A Projective Method for Linear Programming with Box-Type Constraints" Algorithmica (1986) 1, 517-527.

[72] Sakarovitch M. *Linear Programming* Springer-Verlag 1983.

[73] Schrijver A. *Theory of Linear and Integer Programming*, John Wiley & Sons 1986.

[74] Shanno D. F. "Computing Karmarkar Projections Quickly" Mathematical Programming (1988) 41, 61-71.

[75] Stewart G. W. *Introduction to Matrix Computations*, Academic Press 1973.

[76] Thomas G. B. Jr. & Finney R. L. *Calculos and Analytic Geometry*, Addison-Wesley Publishing Company, 1984.

[77] Tinney W. F. & Walker J. W. "Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization" Proceedings of the IEEE (1967) 55, 1801-1809.

[78] Todd M. J. & Burrell B. P. "An Extension of Karmarkar's Algorithm for Linear Programming Using Dual Variables" Algorithmica (1986) 1, 409-424.

[79] Tomlin J. A. "An Experimental Approach to Karmarkar's

Projective Method for Linear Programming" Mathematical Programming Study (1987) 31, 175-191.

[80] Vanderbei R. J. "Karmarkar's Algorithm and Problems with Free Variables" Mathematical Programming (1989) 43, 31-44.

[81] Vanderbei R. J. & Meketon M. S. & Freedman B. A. "A Modification of Karmarkar's Linear Programming Algorithm" Algorithmica (1986) 1, 395-407.

[82] Yannakakis M. "Computing the Minimum Fill-in is NP-Complete" SIAM Journal of Algorithms and Discrete Methods (1981) 2, 77-79.

[83] Ye Y. & Kojima M. "Recovering Optimal Dual Solutions in Karmarkar's Polynomial Algorithm for Linear Programming" Mathematical Programming (1987) 39, 305-317.

[84] Zimmermann U. "On Recent Developments in Linear Programming" Technische Universitat Braunschweig 1986.

[85] Zollenkopf K. "Bi-Factorisation-Basic Computational Algorithm and Programming Techniques" Proceedings of the Oxford Conference of the Institute of Mathematics and its Applications. J.K. Reid Academic Press. (1971) 75-96.

[86] Adler I. & Karmarkar N. & Resende M. G. C. & Veiga G. "Data Structures and Programming Techniques for the Implementation of Karmakar's Algorithm" ORSA Journal on Computing (1989) 1, 84-106.

[87] Anstreicher K. M. "A Combined Phase I - Phase II Projective Algorithm for Linear Programming" Mathematical Programming (1989) 43, 209-233.

[88] Gonzaga C. C. "Large-Steps Path-Following Method for Linear Programming: Barrier Function Method" COPPE 1989.

[89] Gonzaga C. C. "Large-Steps Path-Following Methods for Linear Programming: Potential Reduction Method" COPPE 1989.

[90] Monteiro R. D. C. & Adler I. "Interior Path Following Primal-Dual Algorithms. Part I: Linear Programming" Mathematical Programming (1989) 44, 27-41.

[91] Monteiro R. D. C. & Adler I. "Interior Path Following Primal-Dual Algorithms. Part II: Convex Quadratic Programming" Mathematical Programming (1989) 44, 43-66.

[92] Shamir R. "The Efficiency of the Simplex Method: A Survey" Management Science (1987) 33, 301-334.